Hardware Variability-Aware Duty Cycling for Embedded Sensors

Lucas Wanner, Charwak Apte, Rahul Balani, Puneet Gupta, and Mani Srivastava
University of California, Los Angeles
wanner@cs.ucla.edu, {charwak, rahulb, puneet, mbs}@ee.ucla.edu

***Abstract*—Instance and temperature-dependent power variation has a direct impact on quality of sensing for battery powered, long running sensing applications. We measure and characterize active and leakage power for an ARM Cortex M3 processor, and show that across a temperature range of 20–60°C there is 10% variation in active power, and 14x variation in leakage power.**

We introduce variability aware duty cycling methods and a duty cycle abstraction for TinyOS that allows applications to explicitly specify lifetime and minimum duty cycle requirements for individual tasks, and dynamically adjusts duty cycle rates so that overall quality of service is maximized in the presence of power variability. We show that variability-aware duty cycling yields a 3–22x improvement in total active time over schedules based on worst-case estimations of power, with an average improvement of 6.4x across a wide variety of deployment scenarios based on collected temperature traces. Conversely, datasheet power specifications fail to meet required lifetimes by 7–15%, with an average 37 days short of a required lifetime of one year. Finally, we show that a target localization application using variability-aware duty cycle yields a 50% improvement in quality of results over one based on worst-case estimations of power consumption.

I. Introduction

Energy management methods in embedded systems rely on knowledge of power consumption of the underlying computing platform in various modes of operation. These power specifications are usually derived from the datasheets. Unfortunately, the microelectronic substrate is increasingly plagued by variability, especially in power consumption, both across multiple instances of a system and in time over its usage life. As a result the nominal power specifications are heavily guardbanded (e.g., see [22]) leaving much of the energy potential or sensing quality untapped.

This material is based upon work supported by the NSF under awards # CNS-0905580 and CCF-1029030. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF. Lucas Wanner is supported in part by CAPES/Fulbright grant #1892/07-0.

Variability, thus far, has been largely addressed by process, device and circuit designers, with software designers remaining isolated from it by a rigid hardware-software interface. Recently there have been some efforts at higher layers of abstraction. For instance, software fault tolerance schemes are used to address voltage [36] or temperature variability [11]. Hardware "signatures" are used to guide adaptation in quality-sensitive multimedia applications in [33]. In embedded sensing, [15], [30] propose node deployment methodologies based on the variability in leakage power across different nodes.

Wireless embedded sensing systems employ a variety of power management techniques to achieve system lifetime objectives [35]. A particularly common technique is duty cycling [14], where the system is by default in a sleep state but woken up periodically to attend to pending tasks and events. A higher duty cycle rate typically translates into higher quality of service [44]. A system with higher duty cycle may, for example, sample sensors for longer intervals or at higher rates, increasing data quality. A typical application-level goal is to maximize quality of data through higher duty cycles, while meeting a lifetime goal. While duty cycles in embedded sensing applications range from below 1% in Car-Park management [4] and CargoNet [29], to greater than 50% in VigilNet [20], often the duty cycle ratio is extremely small ($<< 1\%$), and the energy consumed by the platform during the sleep state accounts for almost all ($> 99\%$) of the energy consumption.

In previous work [41], we measured and characterized instance and temperature-dependent sleep power consumption for the Atmel SAM3U, a contemporary embedded processor based on an ARM Cortex M3 core, and discussed the implications of this variation to system lifetime and potential variability-aware software adaptation [42]. The class of low-end 32-bit embedded processors represented by the Cortex M3 is suitable for sensing applications where nodes perform data collection, aggregation, and inferences in a duty cycled fashion, but not for processing intensive, constantly active applications such as video surveillance, which are out-

side the scope of this work. The variations we observed with the SAM3U are comparable to those found in other similar embedded processors [8]. In this paper, we: (i) extend our measuring and modeling of power for the SAM3U, including measurements over a wide temperature range, power decomposition, and characterization of active power variation; (ii) derive projections of sleep and active power for embedded processors with scaling of manufacturing technology; (iii) evaluate variability-aware duty cycle scheduling methods under a variety of deployment scenarios and across projected future variation; and (iv) study the impact of duty cycle to an application's quality of inference. The contributions of our work are the following:

- measurement, characterization, and projections of power variation for embedded processors
- a duty cycle adaptation abstraction for TinyOS, an operating system for embedded sensors
- proposal and analysis of variability-aware duty cycle adaptation methods
- analysis of the impact of duty cycle to quality of sensing.

Our analysis of duty cycle adaptation methods shows that ignoring instance and temperature-dependent variability in low power embedded sensing systems leads to either untapped energy potential, or unmet lifetime requirements. Our duty cycle abstraction for TinyOS allows applications to explicitly specify lifetime and minimum duty cycle requirements for individual tasks and dynamically adjusts duty cycle rates according to a variability-aware scheduler so that overall quality of service is maximized.

The remainder of this paper is organized as follows: Section II presents related work. Section III presents measurement, modeling, and projections of power consumption variability for embedded processors. Section IV discusses software mechanisms to adapt to hardware variation. Section V discusses duty cycle scheduling methods. Section VI evaluates our software adaptation mechanisms under different deployment scenarios, and presents application results that demonstrate the benefits of higher duty cycles to quality of service. Section VII presents our final remarks.

II. Related Work

Prior work that addresses variability can be classified into (i) statistical design approaches [32] [13] [23], (ii) post silicon compensation and correction [18] [25] [39], and (iii) variation avoidance [12] [5] [16]. Our work differs in that it addresses hardware variability in the operating system layer. The closest resemblance is with [33], which proposes adapting software video codec configurations based on hardware signatures. In the context of embedded sensing, our work is closest to [30] [15], which propose sensor node deployment methodologies based on variability in leakage power across different nodes.

Variations in power consumption can be interpreted as changes in resource (energy) usage (and hence availability). Imprecise computation [28] has been explored in the context of energy-aware systems, where tasks may be interrupted, producing an approximate but usable result, according to energy availability and lifetime requirements [10] [43]. Similarly, in energy harvesting, tasks can be adapted to cope with fluctuating energy availability [24].

Several systems have explored the concept of alternative task implementations with different resource usage patterns and quality of service characteristics. Levels is an energy-aware programming abstraction for TinyOS based on alternative tasks [26]. Programmers define task levels, which provide identical functionality with different quality of service and energy usage characteristics. The run-time system choses the highest task levels that will meet the required lifetime. In our work, programmers define tasks with either adjustable periods or iterations. This is similar to adaptive sampling mechanisms used on sensing applications [1]. Our variability-aware adaptation model could support an alternative task abstraction, indirectly adjusting duty cycles by selecting higher or lower levels. Conversely, different levels could represent different duty cycle rates. Per-task energy accounting and distribution is explored in [37], [45].

III. Power Consumption Variability in Modern Embedded Processors

In battery powered embedded sensors it is imperative to understand the relationship between power consumption, quality of sensing, and system lifetime. Power consumed in an embedded class microprocessor chip is broadly classified into active mode and sleep mode. Figure 1 depicts the various components of the total power consumed in a contemporary embedded processor.

A. Experimental Setup

In our experiments, we study the temperature dependence of active and sleep mode power consumption in embedded processors and propose models to characterize it. Our measurements show sleep and active power as a function of temperature across several instances of Atmel SAM3U microcontrollers in LQFP144 packages. While we have been unable to determine from available literature the precise technology node the chip is fabricated

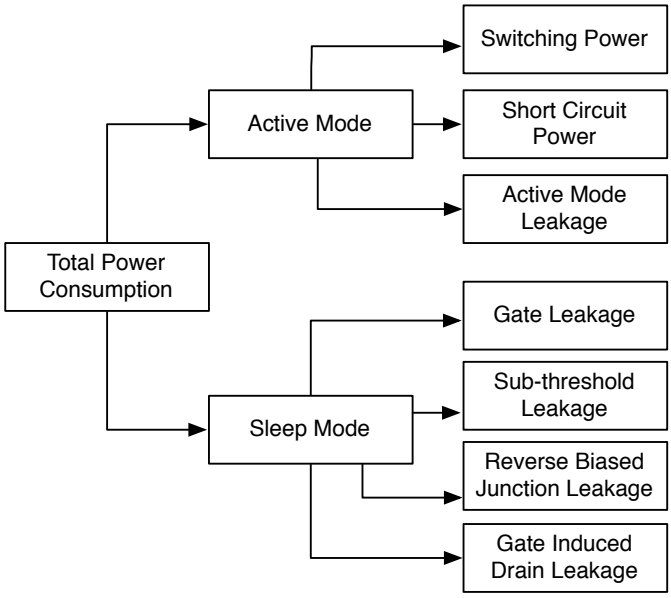
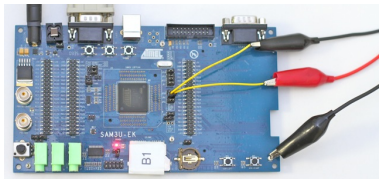
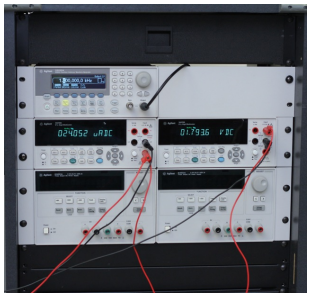


Fig. 1. Power consumption in contemporary embedded processors.



(a) SAM3U-EK Board

(b) Temperature Chamber



(c) Power Measurement Setup

Fig. 2. Experimental Setup

in, indirect evidence as well as the vintage suggests that it is most likely fabricated in a 130 nm process.

For our measurements, we used ten identical SAM3U-EK development boards. These boards feature jumpers that allow power measurements for different components. We measured current and voltage on going into the SAM3U core, with all peripherals except for the real time clock disabled. To obtain synchronized voltage and current measurements we used a pair of Agilent 34410A digital multimeters with a basic accuracy of 0.06%, externally triggered by a function generator. Each measurement point represents the average power dissipated by the core across 50,000 measurements, with a sampling rate of 1,000 samples per second. We used a TestEquity 115F temperature chamber allowing control of ambient temperature with ± 0.5°C accuracy. As discussed in section III-B1, core temperature is effectively equivalent to ambient temperature for the SAM3U in a LQFP144 package. Figure 2 illustrates our test setup.

B. Sleep Mode Power Consumption

With shrinking geometries the ratio of sleep mode power to active mode power has been increasing (as high as 40% in chips fabricated using 65nm technology) [34]. This is due to the inability to turn devices "off" effectively as device dimensions continue to shrink. Manufacturing spread in transistor parameters can cause up to 20x variation in sleep mode power [6] in addition to substantial variation with supply voltage and temperature. Specifically in context of wireless sensor platforms, which often are deployed in extreme ambient conditions, the variation in leakage power during the lifetime of a device may be substantial.

1) Analytical Modeling of Sleep Power : Static power has four main sources: (i) sub-threshold leakage current that flows between source and drain of a MOSFET for gate-to-source voltages below the threshold, (ii) gate leakage current due to tunneling of carriers through the gate oxide to the substrate, (iii) reverse-biased junction leakage current which flows from the source/drain regions to the substrate through the reverse biased p-n junctions due to band-to-band tunneling and diffusion, and (iv) gate-induced drain leakage current due to band to band tunneling in the region of overlap between the gate and drain. At temperatures below 150°C, only the first two components are large enough, and only sub-threshold leakage exhibits strong variability with temperature. Therefore, sleep power can be modeled as the following function of temperature (derived from [7]):

$$P_{sleep} = V_{dd}(AT^2 e^{-B/T} + I_{gl}) \quad (1)$$

where A and B are technology-dependent constants, I_{gl} is the temperature-independent gate leakage current, and T is the core temperature. Coefficients in the model are fitted to individual instances, and hence capture both temperature and instance-dependent variability. We combine the sleep power model with a model of the thermal dynamics of a packaged chip [21]:

$$RC\frac{dT(t)}{dt} + T(t) - RP(t) = T_{amb} \quad (2)$$

where $T(t)$ and $P(t)$ are the core temperature and power consumption of the chip at time t, R and C are the thermal resistance and capacitance of the chip package, and T_{amb} is the ambient temperature. At steady state $\frac{dT(t)}{dt} = 0$, so that $T_{steady-state} = T_{amb} + RP(t)$.

For the SAM3U in a LQFP144 package, the typical values of R and C are 50°C/W and 4–5 J/°C respectively. The nominal static power of SAM3U is 30 μW. Nominal active power when operating at 4 MHz while performing a Dhrystone benchmark is 9 mW. From (2), when sleep mode power measurements are performed, the self-heating of the chip due to static power consumption is negligible, and in active mode the temperature difference between ambient and core temperature is ∼0.5°C.

2) Experimental Measurements: based on the preceding analysis, it is reasonable to assume that the static power follows a similar dependence on ambient temperature as given by (1). We verify this assumption through measurements and characterize each instance of microcontroller based on the above model.

For leakage measurements, we disable all peripheral devices in the SAM3U except for the Real-Time Clock (RTC), select the chip's internal 32kHz RC oscillator as clock source, configure the chip for wakeup with an RTC interrupt, and execute the "wait for event" instruction, which causes the processor to enter *sleep* mode. In addition to *sleep*, the SAM3U microcontroller features two other low power modes. The first, *backup*, completely powers off the core. While this results in the lowest possible power consumption, it also results in wakeup times more than 50 times larger than in other modes, and therefore is not practical for duty-cycled systems. The second low-power mode, *wait*, allows fast wakeup with some specific clock configurations and wakeup sources. In our measurements, we found power dissipated in *wait* mode to be equivalent to power in *sleep* mode with the aforementioned configuration.

Figure 3 shows the experimental data for sleep power consumption of the SAM3U instances across a temperature range fitted to the analytic model discussed earlier, using minimum mean square error criterion. As expected, individual processor instances exhibit large sleep power variations over the temperature range. While change in sleep power for any individual processor is monotonic, the magnitudes of variations are different so that relative rankings of different processors change over temperature. Root mean square error between measurements and model across all instances was 6.7 μW. Over a temperature range of 20–60°C, which is representative of the temperatures that embedded sensors deployed under unregulated and extreme ambient conditions often face (e.g. in factories, desert, etc.), total variation across all ten instances was 14x.

C. Active Mode Power Consumption

1) Switching Power: Switching power is consumed when devices switch between different logic values. The

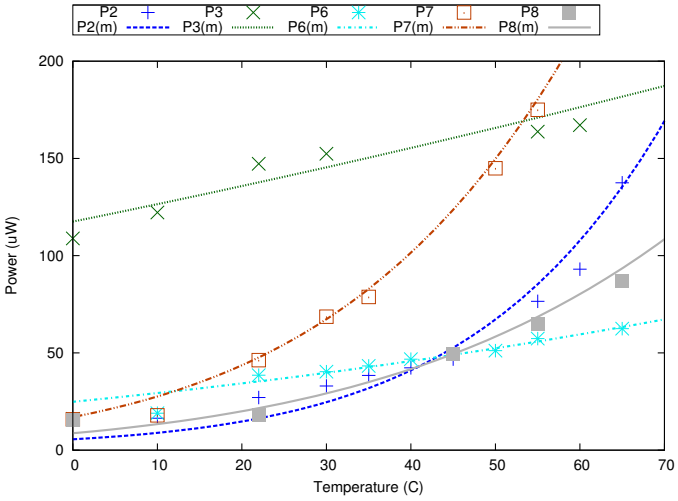


Fig. 3. Comparing measured vs. modeled variability of sleep power with temperature. Only five of the ten instances are shown for clarity.

analytical expression of switching power is,

$$P_{switching} = \alpha C V_{dd}^2 f \tag{3}$$

where α is the activity factor that represents how often a gate switches per clock cycle, C is the capacitance at the switching node, V_{dd} is the supply voltage and f is the clock frequency.

The temperature dependence of switching power is primarily due to the capacitance. The switching node capacitance has three components (i) Gate capacitance, (ii) Wire capacitance and (iii) Diffusion capacitance. As per [17], the gate and diffusion capacitance components have a small linear temperature dependence. In the range of interest, the energy spent in charging this temperature dependent capacitance increases by 8% in a temperature range from 223K to 393K. The interconnect capacitance is largely dependent on the the physical dimensions of the wires and permittivity of the inter-layer dielectric. The physical dimensions and dielectric both have negligible temperature coefficients for our purposes.

Interestingly, for the SAM3U processors we used, even the clock frequency varied by 2% across the temperature range and 6% across different instances as shown in Figure 4. To measure clock frequency, we toggled an I/O pin at a rate proportional to the core frequency, and observed the resulting frequency at different temperatures with a digital oscilloscope. All processors were set to operate nominally at 4 MHz. The clock is generated by an internal ring oscillator (RO). Generally, RO frequency decreases with an increase in temperature so there is always temperature compensation provided with the RO circuit to generate the clock at the specified frequency across temperature. We observe that the frequency increases with temperature. This is likely due to temperature over-compensation.

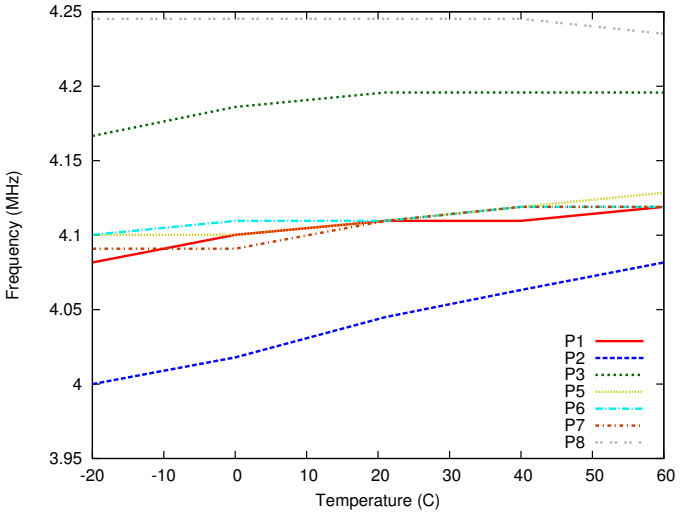


Fig. 4. Measured frequency variation with temperature

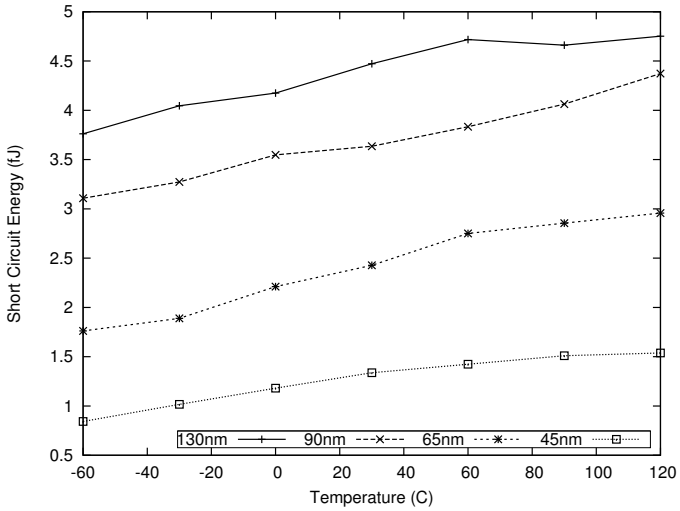


Fig. 5. Temperature dependence of SCE for an inverter chain

2) Short Circuit Power: In CMOS logic, when inputs ramp between logic levels there is a direct path between supply to ground for a short time. Power dissipated during this period is termed as the short circuit power. It contributes to 10% of the total active power nominally at 130nm. HSPICE runs on an inverter chain for 130/90/65/45 nm indicate that this component becomes more prominent as we advance to deep sub-micron technologies. This is primarily because at 45nm node the transistors spend a higher percentage of time in the short-circuit state when the inputs are ramping as compared to when they are at stable logic levels. The short circuit power of an inverter chain varies by as much as 20% at 130nm to 80% at 45nm across $-50°C - 120°C$ in our experiments using IBM and PTM models [19] as shown in Figure 5. The short circuit energy is expressed as:

$$E_{sc_{High-to-low}} = \int V_{dd}.i_p dt \tag{4}$$

$$E_{sc_{Low-to-high}} = \int V_{dd}.i_n dt \tag{5}$$

where i_p and i_n are the drain to source currents of the PMOS and NMOS transistors.

The drive current of a short channel MOSFET is expressed by the alpha power law model [38] and primarily depends on three factors, (i) Mobility (μ), (ii) Overdrive volage ($V_{GS} - V_{th}$) and (iii) Saturation velocity (v_{sat}) The temperature dependence of these three parameters as per [2] is as follows:

$$\mu(T) = \mu(T_0)(\frac{T}{T_0})^m \tag{6}$$

$$V_{th}(T) = V_{th}(T_0) - k(T - T_0) \tag{7}$$

$$v_{sat}(T) = v_{sat}(T_0) - h(T - T_0) \tag{8}$$

where T_0 is the nominal temperature (300 K), $\mu(T_0)$, $V_{th}(T_0)$, and $v_{sat}(T_0)$ are mobility, threshold voltage, and the saturation velocity at the nominal temperature respectively, and k is the temperature coefficient ($\sim$0.8 $mV.K^{-1}$) of threshold voltage. The temperature coefficients m and h are technology dependent. m is ideally 1.5 but can reach unity [2], [9]. The value of h is around 150 $ms^{-1}.K^{-1}$. It follows that these parameters have a close-to-linear dependence on temperature.

As temperature increases, (i) The decrease in V_{th}, increases the overdrive ($V_{GS} - V_{th}$) of the transistor and hence causes an increase in drive current, (ii) The decrease in mobility and v_{sat} result in lowering of the drive current in the linear and saturation regions. The impact of temperature on drive current, delay, and power is dependent on which of the two effects dominates.

V_{dd} is high relative to V_{th} for older technologies and as the temperature coefficient of threshold voltage is low, the increase in the device overdrive is shadowed by mobility degradation in technology older than 180nm. In these technologies, the drive current and delay exhibit monotonic behavior. The high temperature corner is always the slow corner while the low temperature corner is the fast corner. In deep sub-micron technologies, V_{dd} and V_{th} are not very different, resulting in a phenomenon called Inverse Temperature Dependence (ITD). The drive current that increases monotonically as temperature decreases shows inversion and starts decreasing if the temperature is lowered beyond a certain value. This is primarily due to the decrease in overdrive voltage as temperature decreases, which starts shadowing the mobility enhancement. Short circuit power measurements also show that ITD causes the roll-off of the short-circuit power curve to be steeper and starts showing up at -40°C.

3) Active Mode Leakage Power: In the active mode, devices that do not switch also consume power. Leakage power has an exponential dependence on temperature, as elaborated in section III-B1. For our experiments, however, P_{active}/P_{sleep} is nominally ∼300. Hence this component has a small effect only on the high temperature range dependence of the the total active mode power.

4) Analytical Modeling of Active mode Power: In the temperature range of interest, $223 - 393$K, the temperature dependence of active mode power can be explained by dividing the range into three regimes.

(i) Low temperature regime: The contribution of active mode leakage is neglected as at low temperatures. An increase in V_{th} causes exponential decrease in leakage. The capacitance increase is a linear function of temperature and hence contributes to the linear dependence of active mode power. At low temperatures, ITD causes the decrease in active mode power to have a steeper slope as temperature reduces. The relationship with temperature is characterized as:

$$P_d = P_{T_1} + k_1(T - T_1)^{\alpha} \tag{9}$$

where, $0 < \alpha < 1$, $T_1 < T < T_2$, k_1 and α are fitting parameters. $T_1 < T < T_2$ defines the low temperature regime.

(ii) Nominal temperature regime: Short circuit and switching power have a linear dependence in this regime while the active mode leakage is neglected. The relationship with temperature is characterized as:

$$P_d = P_{T_2} + k_2(T - T_2) \tag{10}$$

where, $T_2 < T < T_3$ and k_2 is a fitting parameter. $T_2 < T < T_3$ defines the nominal temperature regime.

(iii) High temperature regime: The switching power varies linearly (capacitance dependence is linear), short circuit power and active mode leakage give active mode power in this regime a super-linear dependence. This can be inferred from the short circuit power and leakage temperature dependence studied in this work. The relationship with temperature is characterized as:

$$P_d = P_{T_3} + k_3(T - T_3)^{\beta} \tag{11}$$

where, $1 < \beta$, $T_3 < T$, k_3 and β are fitting parameters. $T_3 < T$ defines the high temperature regime.

Figure 6 shows the active power model fitted to our measured data for active mode power consumption across a temperature range. For this experiment, we used the same measurement setup as in the sleep mode power measurements. All SAM3U peripherals were disabled, except for the RTC. The core was clocked from the internal ring oscillator at 4MHz, continuously running a Dhrystone benchmark program. Root mean square error between measurements and model across all instances was 0.02 mW. Over a range of 20–60°C, total variation across all ten instances was 10%.

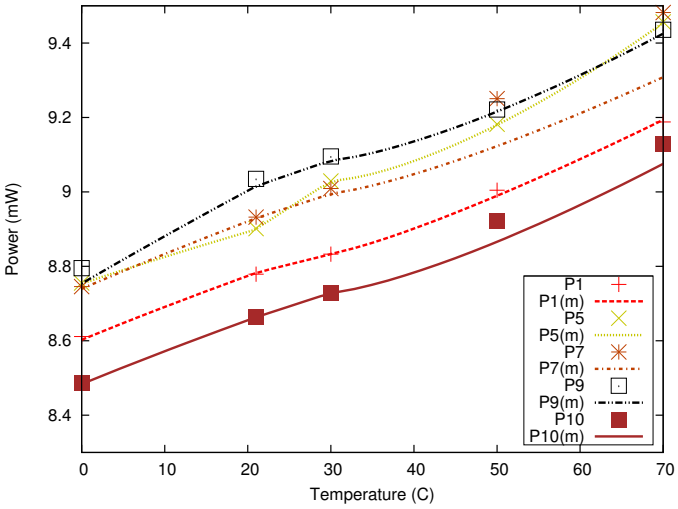


Fig. 6. Measured data and the active power consumption model. Only five of the ten instances are shown for clarity.

D. Projecting temperature dependence of power for advanced technologies

The characterization discussed above was based on actual measurements from hardware manufactured in 130nm. Nominal power consumption variability is characterized in ITRS across sub-130 nm technology nodes. We use this data to project the spread of power vs temperature curves. PTM 130, 90, 65 and 45 nm spice models were used to project the scaling of active and sleep mode power across these technology nodes based on a ring oscillator design. Relative temperature dependence was assumed to be the same as observed in the measurements. We assume that the temperature regimes remain constant across technologies. The projections can be refined by calibrating these regimes for advanced technology using real measurements. We also assume in our experiments that while in active mode 20% of devices are switching. Tables I and II list typical sleep and active mode power model parameters at each technology node. Figures 7 and 8 show expected projection of nominal sleep and active power across temperature resulting from these parameters for each node technology.

TABLE I
SLEEP POWER MODEL PARAMETERS ACROSS TECHNOLOGIES

Instance	A	B	I_{gl}	V_{dd}
Typ_130	1.0	2605.5	0.0	1.8
Typ_90	1.26	2400	0.2	1.8
Typ_65	1.76	2300	0.6	1.8
Typ_45	2.52	2100	1	1.8

TABLE II
ACTIVE MODE POWER MODEL PARAMETERS ACROSS TECHNOLOGIES

Instance	k_1	k_2	k_3	a	b	$P_{T1}(mW)$	$P_{T2}(mW)$	$P_{T3}(mW)$	T_2 (°C)	T_3 (°C)
Typ_130	0.0511	0.0095,	0.003325	0.669	1.3456	7.7	8.6	8.68	21	30
Typ_90	0.0246	0.0046	0.0016	0.71	1.5228	2.82	3.33	3.37	21	30
Typ_65	0.0094	0.00175	0.000615	0.735	1.6335	0.9	1.125	1.14	21	30
Typ_45	0.0046	0.000855	0.0003	0.755	1.722	0.364	0.479	0.4867	21	30

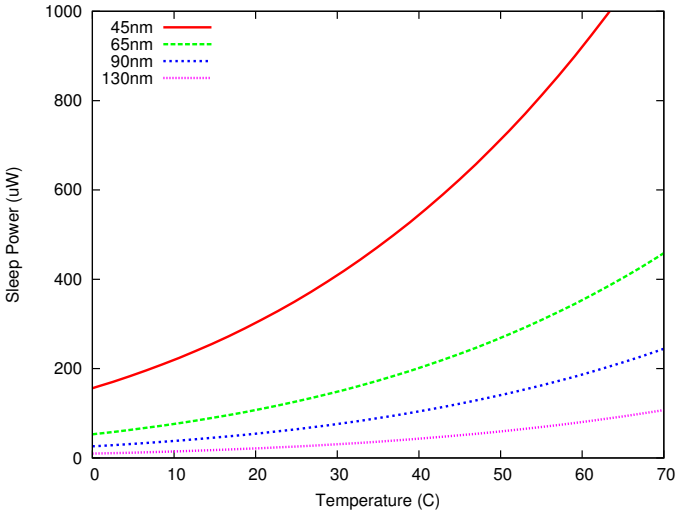


Fig. 7. Sleep power projection across technologies.

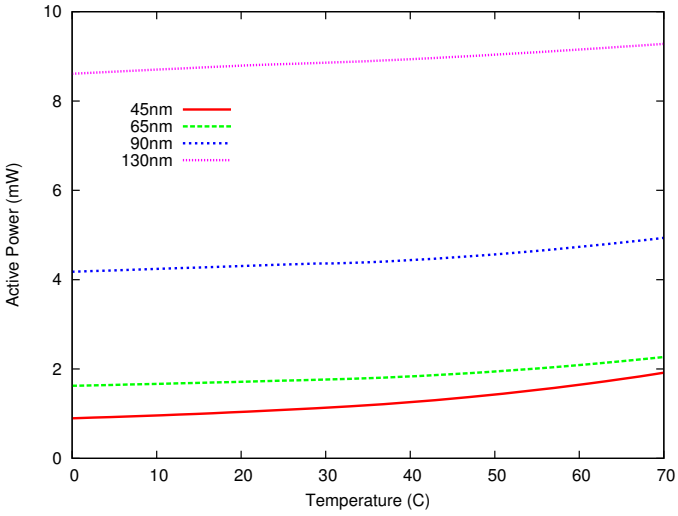


Fig. 8. Active power projection across technologies.

IV. VARIABILITY-AWARE SOFTWARE ADAPTATION

As the discussion in the preceding section shows, significant variability in power across nominally identical instances and across temperature is already present in contemporary embedded processors. Duty cycling is particularly sensitive to variations in sleep power at low duty cycling ratios. Variability implies that any pre-deployment choice of duty cycle ratio that is selected to ensure desired lifetime needs to be overly conservative and result in lower quality of sensing or lifetime.

In order to maximize the sensing quality in the presence of power variation, an opportunistic sensing software stack can help discover and adapt the application duty cycle ratio to the sleep mode power variations across parts and over time. The run-time system for the opportunistic stack will have to keep track of changes in hardware characteristics and provide this information through interfaces accessible to either the system or the applications. Figure 9 shows several different ways such an opportunistic stack may be organized; the scenarios shown differ in how the sense-and-adapt functionality is split between applications and the operating system. Scenario 1 relies on the application polling the hardware for its current "signature". In the second scenario, the application handles variability events generated by the operating system. In the last scenario, handling of variability is largely offloaded to the operating system.

Using architecture similar to that of scenario 3 in Figure 9, we have implemented a prototype variability-aware duty cycling framework in TinyOS. Application modules specify to the scheduler a range of acceptable duty cycling ratios, and the scheduler selects the actual duty cycle based on run-time monitoring of operational parameters, and a power-temperature model that is learned off-line for the specific processor instance. While this approach is potentially less flexible than the ones presented by scenarios 1 and 2, it simplifies application development by abstracting the underlying complexities of the variability signature model.

TinyOS [27] differs from traditional operating systems in that it is *event-based*. Applications respond to events (e.g. interrupts from hardware, incoming radio messages) with event handlers. Handlers typically complete within a few hundred processor cycles. To execute long running computations, applications post tasks, which work as deferred function calls. Whenever the system has no tasks to schedule, it puts the processor in sleep mode, waiting for the next interrupt which will trigger new event handlers, and potentially new tasks. The event handler / background tasks model of TinyOS naturally lends itself to duty cycled systems: event handlers and tasks represent active periods, empty scheduler queues lead to inactive periods. Nevertheless, there's no explicit support for discovering and adapting duty cycle in TinyOS.

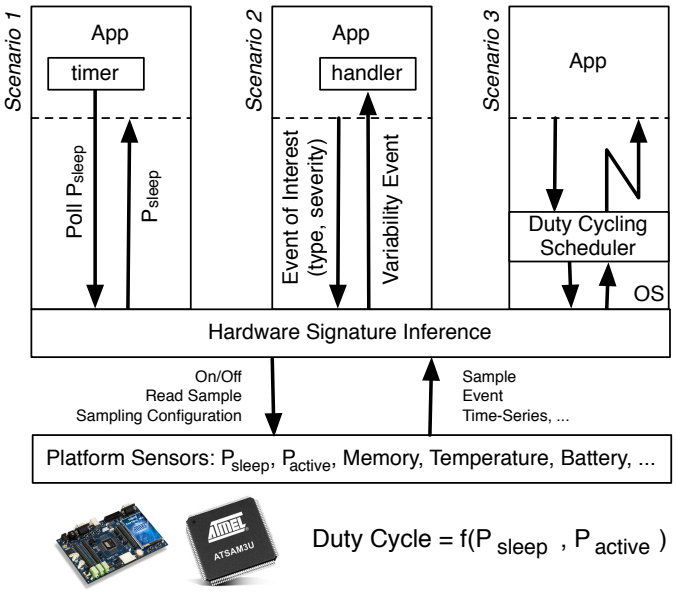


Fig. 9. Designing a software stack for variability-aware duty cycling

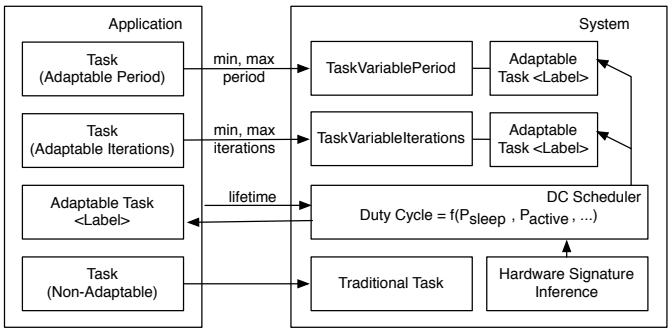


Fig. 10. System architecture for variability-aware duty cycle scheduling in TinyOS

We introduce a new *Duty Cycle Scheduler* to TinyOS. Figure 10 shows our system architecture. A hardware signature inference module provides power vs. temperature curves for each processor instance. While in our work we assume that these curves are pre-characterized, extensions to this module could feature online learning through dedicated power meters and take other variability vectors such as aging into account.

The scheduler determines an allowable duty cycle based on: (i) sleep and active power vs. temperature curves, (ii) temperature profile for the application, which can be pre-characterized or learned dynamically, (iii) lifetime requirement, (iv) battery capacity, and (v) the variability-aware duty cycle formulation in Eq. (14).

To maintain compatibility with existing TinyOS tasks, we introduce a new class of *Adaptable Tasks*. These tasks respond to events from the Duty Cycle scheduler that inform them of their current and allowable duty cycle. However, the system does not *enforce* adaptation of these tasks. These are assumed to adapt according to the duty cycle change event from the scheduler. Adaptable tasks also allow for some of the flexibility present in scenario 2 in Figure 9. A module using adaptable tasks could, for example, use an alternative function mechanism such as the one in Levels [26].

For standard applications, we provide two additional classes of tasks which implement two common adaptation scenarios: tasks with variable iterations and tasks with variable period. For the first class, the programmer provides a function that can be invoked repeatedly a bounded number of times within each fixed period. For the second class of tasks, the application programmer provides a function representing task functionality that is invoked once within each variable but bounded period of time. Internally, each of these tasks uses an adaptable task and unique identifier. The system adjusts the number of iterations or period of the task based on the allowable duty cycle informed to its underlying adaptable task.

To adapt duty cycle, the system needs to account active and sleep time. Active time is divided into non-adaptable computation time C_f (for traditional tasks and interrupt handlers) and adaptable computation time $C_a(i)$ (for each adaptable task i). For each task activation, the system registers timestamps and accumulates computation time in the appropriate counters. This incurs in a small overhead to task activations quantified in section VI-F.

For every accounting period τ, the system compares total computation time $C_\tau = C_f + \sum_1^N C_a(i)$, where N is the number of adaptable tasks in the system, and allowable active time $C_{DC} = \tau \times DC$, where DC is the allowable duty cycle for the node. To allow adequate timing estimations with low duty cycles, the accounting period τ is large enough to encompass several active/sleep cycles for each task ($\tau = 10$ minutes in our implementation). If $|C_\tau - C_{DC}| <= \delta$, where δ is an arbitrary tolerance, the system has converged to the allowable duty cycle. Otherwise, each adaptable task is assigned a new allowable computation time $C_{DC}(i)$ obtained by dividing available active time equally between all adaptable tasks. The new computation time is informed to the tasks in the form of a ratio to the previous time $C_a(i)$ through an event. The ratio is used to adjust the number of iterations or period duration of tasks.

While we use a simple scheduling mechanism, typical of small embedded sensor operating systems, we could easily incorporate other priority-based time distribution schemes in our scheduler, e.g. as explored in [37], [45]. For example, instead of receiving equal time slots, all tasks could be adapted proportionally with a system-wide ratio. Likewise, a reward-based activation and time distribution mechanism, e.g. [3], could prioritize critical tasks and selectively discard "less important" tasks in order to meet the allowable duty cycle.

V. Duty Cycle Scheduling

A duty cycle schedule indicates the activity rate of a system at any point in its lifetime. An optimal duty cycle schedule maximizes the active time of the system across its desired lifetime, given an energy constraint. If there is no variability in power consumption, the optimal duty cycle schedule can be uniform across the lifetime of the system. Given an energy budget of E Joules, a lifetime of L seconds, and invariable constants for active and sleep power consumption P_A and P_S Watts, the maximum allowable allowed duty cycle DC is given in (12). The values of P_A and P_S can be typically obtained from the processor datasheet. We henceforth refer to this as ***datasheet-based duty cycle***.

$$P_A \cdot DC + P_S \cdot (1 - DC) = \frac{E}{L}$$
$$DC = \frac{E - L \cdot P_S}{L \cdot P_A - L \cdot P_S} \tag{12}$$

A. Variable Power Consumption

When instance and temperature-dependent variation is taken into consideration, the ***worst-case uniform duty cycle*** can be found by applying the worst-case active and sleep power consumption across all instances and operating temperature range as constants P_A and P_S in (12). We henceforth refer to this as ***worst-case DC***.

With prior characterization, active and sleep power can be expressed as functions of temperature $P_A(T)$ and $P_S(T)$. Additional peripheral components used while in active mode, such as radios or sensors, can be added to total active mode power as constants or functions, depending on whether power variation is present or not. In our preliminary experiments, we did not find any significant variation in power consumption across instances and temperature in peripheral components. If the temperature profile is known (or can be learned) for the lifetime of the system, temperature can be expressed as a frequency distribution. For a known operating temperature profile and a given processor instance, the problem of finding an ***optimum duty cycle*** can be formulated as a linear program. Given the expected frequency distribution of (discretized) temperatures across the lifetime of the application, the optimum duty cycle at each temperature T, DC_T is given by (13):

$$\arg\max_{DC_T} \sum_{T=T_{min}}^{T_{max}} DC_T f_T \tag{13}$$

$$\text{s.t.} \sum_{T=T_{min}}^{T_{max}} f_T \cdot (P_A(T) \cdot DC_T + P_S(T) \cdot (1 - DC_T)) \leq \frac{E}{L}$$
$$DC_{min} \leq DC_T \leq DC_{max}$$
$$T_{min} \leq T \leq T_{max}$$

where f_T is the relative frequency of temperature T across the lifetime L, assuming discretized temperature bins. DC_{min} and DC_{max} are the minimum and maximum duty cycles allowed for the application. The maximum duty cycle constraint can be used to limit duty cycles when increasing duty cycle beyond a given rate would bring no further increase to quality of service.

B. Variability-Aware Uniform Duty Cycle

Assuming a uniform duty cycle $DC_T = DC^*$ independent of temperature, we can determine DC^* that satisfies the constraints given in (13).

$$DC^* = \min\left[\gamma, DC_{max}\right] \tag{14}$$
$$\text{where } \gamma = \frac{E - L \cdot \sum_{T=T_{min}}^{T_{max}} P_S(T) \cdot f_T}{L \cdot \sum_{T=T_{min}}^{T_{max}} (P_A(T) - P_S(T)) \cdot f_T}$$

Moreover, it can be shown that when $P_A(T) - P_S(T)$ is constant across all T, DC^* is the ***uniform duty cycle*** that optimizes the linear program in (13). We observed this to be *practically* true under nominal operating temperatures for the current generation microprocessors, like the Atmel SAM3U, because (i) their sleep power consumption $P_S(T)$ is much less than active power consumption $P_A(T)$, and (ii) the $P_A(T)$ is effectively constant as active mode leakage power is insignificant for their fabrication technology, and switching power variation across normal temperatures is small. Henceforth, whenever we refer to ***variability-aware duty cycle***, we are referring to (14).

C. Reactive Duty Cycle

Allowable duty cycle rates can also be found dynamically through measurements or estimations of past power consumption, given total energy capacity at the start of lifetime. Energy consumption can be directly measured with dedicated monitors [31], inferred from remaining battery capacity [26], or through variability-aware models that estimate energy expenditure by measuring conditions that affect power consumption, e.g. temperature and activity rates.

In a reactive model, duty cycle can be dynamically determined at time t as a ratio of duty cycle at time $t-1$, according to energy spent from time $t-1$ to time t, and remaining energy in the system. Remaining energy at time t is given by $E_t = E - \sum_{i=0}^{t-1} P_i$, where E is the total energy capacity and P_i is power estimated or measured at time i. An example of a ***reactive duty cycle*** adaptation model is given in (15).

$$DC_t = \frac{E_t \cdot DC_{t-1}}{(E_t - E_{t-1}) \cdot (L - t)} \tag{15}$$

The reactive model in (15) assumes that the power consumption rate for the previous time period is indicative of the power consumption for the remainder of lifetime of the system. While more complex models could incorporate longer histories, any reactive model will depend on accurate measurement of past energy consumption or estimation of remaining battery energy. While systems with dedicated power monitors have been explored in the literature [31], it is acknowledged that their integration in low power sensing platforms would likely result in prohibitive cost overhead. Hence, most systems rely on estimations of remaining battery capacity to infer energy consumption [26]. Battery capacity estimation is a research issue in itself and subject to inaccuracies. In target systems with long lifetimes (e.g. greater than a year), in which the energy consumed in one hour might be less than 0.01% of total battery capacity, this makes short term adaptation problematic. We therefore do not use this method.

VI. Evaluation

A. Evaluation Scenario

To evaluate the duty cycle optimization methods, we make use of a common scenario in embedded sensing: a long running duty cycled application with a limited energy source (battery), which periodically becomes active to perform sensing/processing tasks, and subsequently returns to a low-power sleep mode until the next period.

We assume an application which, when active, uses only the main processor running at 4MHz, and when in sleep mode, disables all peripherals except for a low-power wake-up timer. Table III summarizes the results we present in this section. Active and sleep power are obtained from the characterization model and measurements presented in Section III. Figure 16 is based on technology projections. All other results are based on the models fitted to SAM3U measurements. The temperature profile is based on hourly temperature data from the National Climactic Data Center [40], and specified for each result.

TABLE III
Summary of Results

Fig.	Type of Result	Variable
11	Duty Cycle Schedules	Time
13	Improvement with Var-Aware DC	Temperature profile
14	Lifetime with Datasheet-Based DC	Temperature profile
15	Improvement with Var-Aware DC	Battery Capacity
16	Improvement with Var-Aware DC	Technology
17	Localization Error	Duty Cycle
18	Localization Error	Time

B. Comparison of Duty Cycle Scheduling Methods

We first compare duty cycle schedules resulting from the worst-case, datasheet-based, and variability-aware methods discussed in Section V. We assume an energy supply of 5400 mAh from two AA batteries, a lifetime of one year, and temperature profile based on the location of Stovepipe Wells, CA (Death Valley National Park), which has extreme seasonal and daily temperature variations and hence clearly illustrates the differences between the duty-cycling regimes.

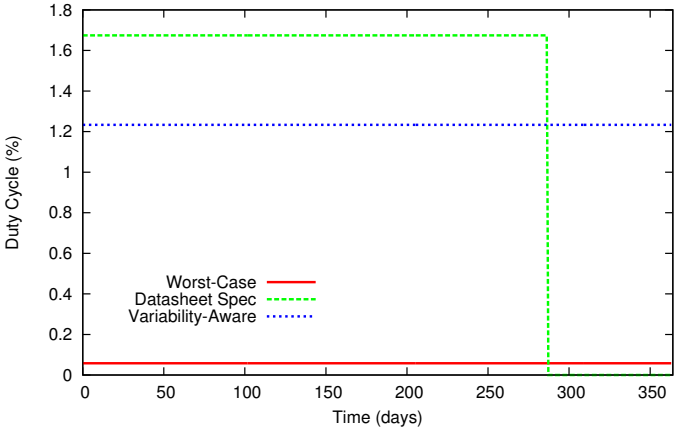


Fig. 11. Schedules from different DC regimes for instance P7.

TABLE IV
Results from DC regimes across all instances

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	Avg.
Improvement of variability-aware DC over worst-case DC (x)										
28	29	6	21	23	26	21	28	9	30	22.2
Energy left untapped by worst-case DC (%)										
81	84	15	60	67	75	61	81	24	85	63
Lifetime reduction with DC based on datasheet (%)										
2.5	0	41	22	16	7	21	4	38	0	15

Figure 11 shows the DC schedules across the lifetime of an application for a single instance (P7). It shows that the duty cycle resulting from from datasheet power specifications does not meet the required lifetime, as the specification is not guardbanded enough. Determining a completely pessimistic sleep power specification is very difficult since leakage distribution has a long tail. The worst-case duty cycle is exceedingly low, as it assumes the worst-case and power across all instances and the entire temperature range to which the application experiences, and hence leaves out untapped energy resources. The variability-aware duty cycle maximizes active time, constrained by lifetime requirements and application temperature profile. Table IV summarizes the results from the various regimes for all instances. On average, we found a 22x improvement in active time with the variability-aware DC over the worst-case DC, 63% of energy potential left untapped by the worst-case DC, and 15% reduction in lifetime with DC based on datasheet specifications.

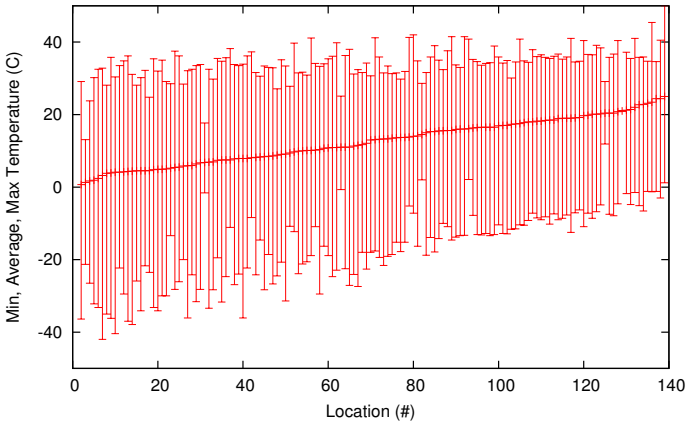


Fig. 12. Temperature profile for test locations.

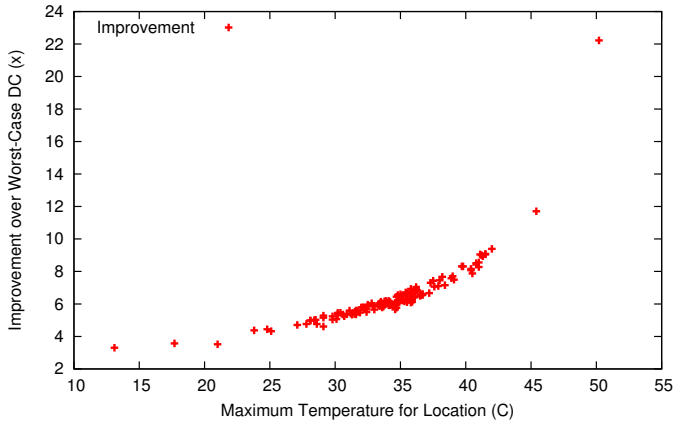


Fig. 13. Improvement over worst-case Duty Cycle for test locations.

C. Temperature Profile

Next, we use the same energy and lifetime scenario described above for 140 locations with different temperature profiles. Figure 12 shows the average, minimum, and maximum temperature for all the test locations.

Figure 13 shows the average improvement of variablity-aware duty-cycle compared to worst-case duty cycle. There is an exponential relation of improvement with temperature. This results from the exponential nature of leakage power with temperature: as maximum temperature increases, leakage power increases exponentially, and hence the worst-case duty cycle is exponentially worse than the variability-aware duty cycle. For any given maximum temperature, improvement also depends on temperature distribution: temperature profiles with higher maximum temperatures benefit more from the variability-aware scheme, as the worst-case power becomes progressively worse with higher temperatures. The average improvement across all temperature profiles is 6.4x.

Figure 14 shows lifetime reduction in days when duty cycle is determined based on the datasheet specification. There is a linear dependence between lifetime and average temperature. As average temperature increases, the difference between actual and spec power increases, and hence lifetime decreases. Across all temperature profiles there is an average lifetime reduction of 37 days for a lifetime of one year.

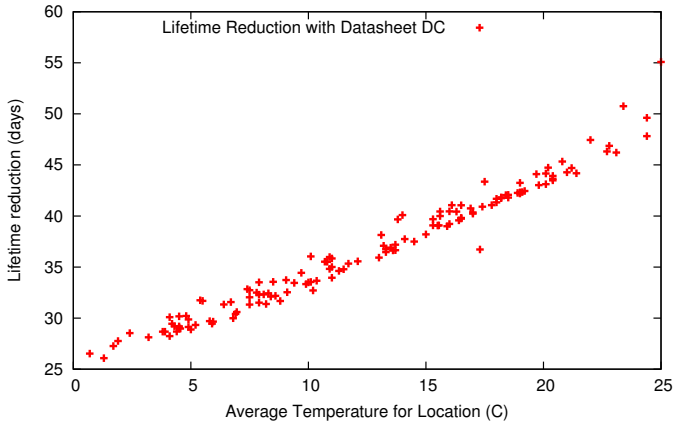


Fig. 14. Lifetime reduction with DC based on datasheet.

D. Battery Capacity

Figure 15 shows the average percentile improvement of a variability-aware duty cycle schedule (14) over the worst-case duty cycle for different battery capacities. Each curve shows improvement with a different temperature profile. Death Valley, CA has the most extreme temperatures, and hence greater improvement. Mauna Loa, HI has low average temperatures and little temperature variation, and hence benefits the least from the variability-aware scheme. Williams, AZ represents the average case.

This plot shows that the variability-aware duty cycling regime is more advantageous for applications with smaller duty cycles. For "small" batteries (5400 mA-h, or 2 AA batteries), improvement is more than 100% for all temperature profiles. The improvement for very large batteries (20 A-h), improvement is between 20% and 30%, depending on temperature profile. The worst-case duty cycle with a 20 A-h battery is more than 5% for all temperature profiles. This suggests that, for current embedded fabrication technologies, this scheme is beneficial for small (less than 5%) duty cycles.

E. Technology Projections

Figure 16 shows the improvement of the variability-aware duty cycle over worst-case duty cycle with technology scaling as per the model presented in section III. As with the previous result, we show three curves with different temperature profiles. For each point in the curve, we simulate a battery capacity large enough to support a worst-case duty cycle of 5%. As noted in the

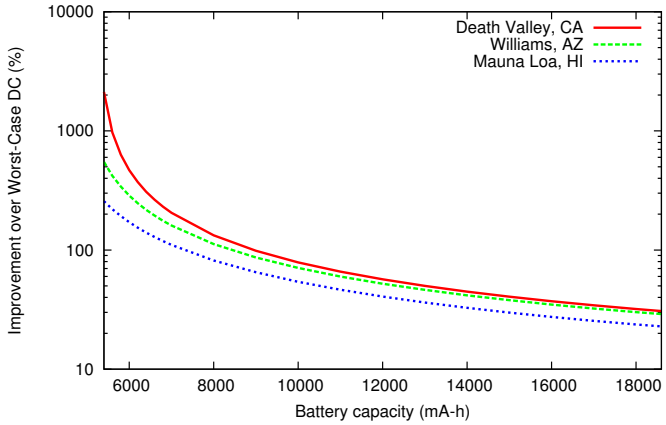


Fig. 15. Improvement over Worst-Case DC across battery capacities.

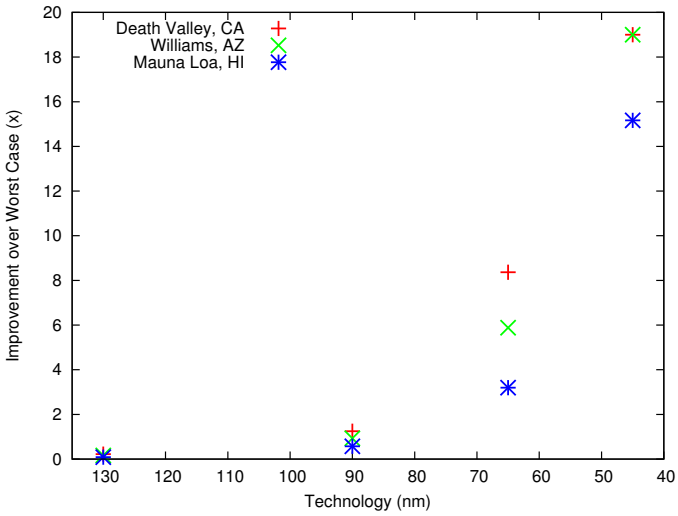


Fig. 16. Projection of improvement over worst-case duty cycle with scaling of technology.

previous results, with a worst-case duty cycle of 5%, there are only marginal benefits of a variability-aware duty cycle schedule for current technology characteristics (130nm). As technology progresses and the ratio between active and sleep power decreases [34], variability-aware duty cycling regime shows considerable benefits even for this relatively high duty cycle. At 45nm, the improvement saturates at 19x for two of the temperature profiles (duty cycle cannot be higher than 100%).

F. Runtime Overheads

We profiled our duty cycle scheduler implementation for the SAM3U processor, running at 4 MHz. Compared to the base TinyOS scheduler, our implementation requires an additional 20 bytes of RAM memory. Each adaptable task instance uses 5 bytes of memory, and the TaskVariablePeriod and TaskVariableIterations abstractions require an additional 4 bytes per task. This overhead is well within the capacity of low-end sensor nodes (typically $>$ 4KB RAM).

Compared to basic TinyOS tasks, each adaptable task activation has an overhead of 10μs. As a point of comparison, the typical startup and conversion times for the ADC in this platform are 30μs and 20μs, respectively. Finally, the uniform variability-aware duty cycle control module, which runs periodically every 10 minutes, completes within $20(N+1)\mu$s, where N is the number of adaptable tasks in the system. This time is required to distribute available active time across all adaptable tasks, and to determine the rate of activity (i.e. period, number of iterations) of each task. When the system becomes stable, i.e., when all the tasks reach their allowable duty-cycle, the uniform variability-aware duty cycle control module completes within $20\mu s$. The runtime overhead of our simple duty cycling abstractions is comparable to related solutions [26], [45].

G. Application Results

Higher duty cycles allow the sensors to stay "on" for a longer time and capture more data during deployment. This typically increases accuracy and shortens response times in high fidelity real-time sensing tasks such as object localization and tracking [44]. For instance, Figure 17 quantifies the effect of different duty cycles on the accuracy of sound source localization with a network of 20 acoustic (e.g. microphone) sensor arrays using Maximum Likelihood Estimation (MLE). It shows that with increasing duty cycles, the performance of the application improves as the mean localization error decreases. The application estimates the location of a target based on line-of-bearing (LoB) measurements from the sensor arrays deployed randomly in a 10m $\times$ 10m field. Each sensor requires approximately 1 second on an embedded ARM-based processor to compute one LoB measurement from raw audio samples. The error in sensor measurements is assumed to be less than 10%.

Moreover, Figure 17 demonstrates that the error also decreases with time, for a specific DC, as the algorithm waits to collect measurements from all the sensors before fusing them to obtain the location estimates. This result is important when the sensor nodes follow asynchronous wake-up and sleep schedules to avoid the control message overhead associated with synchronous duty cycling. For instance, the sensors duty cycle asynchronously with a period of 200s in our simulations of the localization application. However, this is based on the implicit assumption that the target remains static during data collection to enable the sensors to obtain relevant LoB measurements. Correspondingly, in Figure 17, the target is assumed to be static for a period of 20s and 40s for the respective plots. This constraint on target motion is due to the limited number of nodes in the simulation

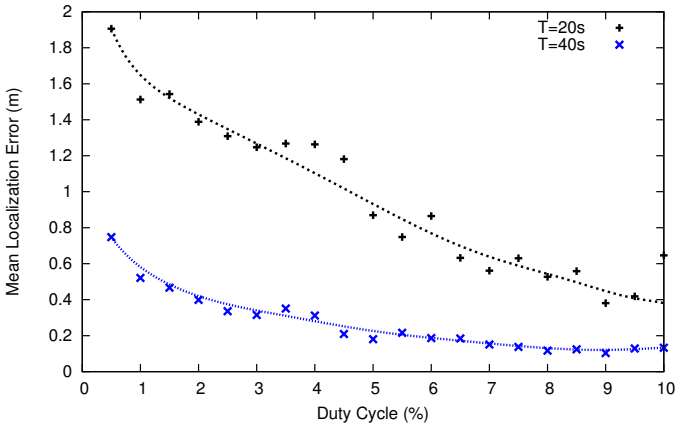


Fig. 17. Mean localization error decreases with increasing duty cycle and time as more data is collected from the sensors. The simulations were adapted from [44] and represent 200s of application execution in real-time. Each point in the plot is generated from an average of 200 simulation runs with varying sensor on/off schedules.

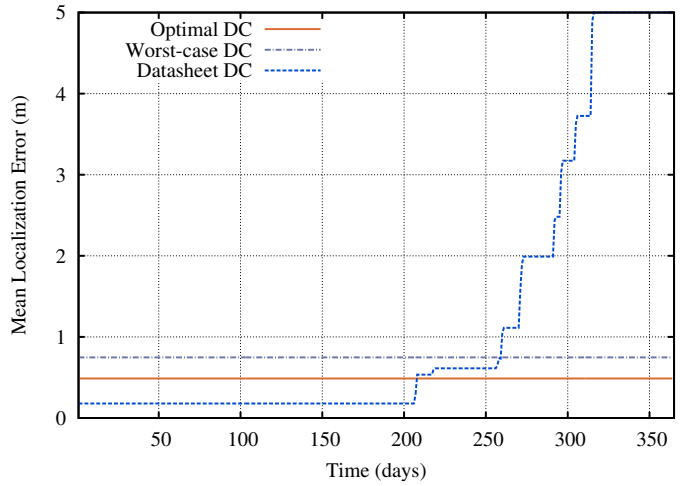


Fig. 18. Mean localization error over lifetime (1 year) of the localization application with variability-aware, worst-case and datasheet-based duty cycle schedules.

scenario, and the low duty cycles in question.

In accordance with these results, Figure 18 demonstrates that the algorithm generates estimates of target location with a lower mean error in presence of variability-aware optimal schemes as compared to worst-case schedules for the same battery capacity and lifetime constraints, using the evaluation scenario described in section VI-B. Although the mean error is the lowest at the start of the deployment with DC schedules determined from datasheet values, it increases steeply towards latter half of the deployment timeline as sensors exhaust their batteries before the intended lifetime of 1 year. In all these simulations, it is assumed that targets appear at the center of the field once every 200s and remain static for a period of 40s at each appearance. As with the previous result, this constraint on target motion could be relaxed if a larger number of nodes was available, or if more energy was available to each node (larger batteries). In the later case, as indicated in Figure 15, the benefits of the variability-aware scheme over the worst-case scheme would be smaller.

VII. Conclusion

In this paper we characterized active and leakage power variation across instances and temperature and showed the implication of power variation to duty cycling in embedded sensors. We presented variability-aware methods to find optimum duty cycle schedules and showed that instance and temperature-dependent variability aware duty cycle scheduling yields a 3–22x improvement in total active time over schedules based on worst-case estimations of power, with an average improvement of 6.4x across a series of deployment scenarios. Conversely, upwards of 63% energy capacity is left untapped when worst-case estimations of power are used to determine duty cycle, and datasheet power specifications fail to meet required lifetimes by an average of 37 days for a required lifetime of one year.

With current technology characteristics, a variability-aware schedule is beneficial for smaller duty cycles ($< 5\%$). With the expected projection of sleep and active power variation with scaling of technology, there will be significant benefits of a variability-aware schedule even for higher duty cycles.

Some classes of embedded sensing applications are not amenable to the type of adaptation described in this work. These include highly synchronized, real-time, or constant data acquisition tasks. Furthermore, our adaptation scheme adds some complexity to the application, in the form of bounds to task activations, which may in turn lead to further complexities in data storage, inference and communication strategies. Nevertheless, we believe that the benefits of our scheme outweigh the added complexity for a large class of sensing applications.

While our duty cycle adaptation scheme indirectly leads to a form of energy-based load balancing across a network of sensors, we do not provide other network-wide adaptation mechanisms such as role selection for nodes, where a node could take different roles (e.g. data collector, router, aggregator) depending on its respective energy rank in the network. We intend to explore this in the future. Ongoing work is also attempting to address issues that may arise from variability in energy availability and in peripheral components. While in our work we relied solely on pre-characterization of variation, in the future a combination of pre-characterization and online learning of variability will allow adaptation to dynamic variation due to aging. Code and data supporting this paper is available at http://variability.org/.

REFERENCES

[1] C. Alippi, G. Anastasi, C. Galperti, F. Mancini, and M. Roveri. Adaptive sampling for energy conservation in wireless sensor networks for snow monitoring applications. In *MASS*, pages 1–6, 2007.

[2] Narain Arora. *Mosfet Modeling for VlSI Simulation: Theory And Practice*. World Scientific Publishing Company, 2007.

[3] H. Aydin, R. Melhem, D. Mosse, and P. Mejia-Alvarez. Optimal reward-based scheduling for periodic real-time tasks. *IEEE Transactions on Computers*, 50(2):111 –130, feb 2001.

[4] J. P. Benson et al. Car-park management using wireless sensor networks. In *IEEE Conf. on Local Computer Networks*, 2006.

[5] S. Bhunia, S. Mukhopadhyay, and K. Roy. Process variations and process-tolerant design. In *Intl. C. on VLSI Design*, 2007.

[6] S. Borkar et al. Parameter variations and impact on circuits and microarchitecture. In *Design Automation Conf. (DAC)*, 2003.

[7] BSIM. http://www-device.eecs.berkeley.edu/∼bsim3/.

[8] D. Bull et al. A power-efficient 32 bit arm processor using timing-error detection and correction for transient-error tolerance and adaptation to pvt variation. *IEEE J. of Solid-State Circuits*, 46(1):18–31, 2011.

[9] Andrea Calimera, R. Iris Bahar, Enrico Macii, and Massimo Poncino. Temperature-insensitive dual-vth synthesis for nanometer cmos technologies under inverse temperature dependence. *IEEE Trans. VLSI Syst.*, 18:1608–1620, 11 2010.

[10] L.N. Chakrapani et al. Ultra-Efficient (Embedded) SOC Architectures based on Probabilistic CMOS (PCMOS) Technology. In *DATE*, 2006.

[11] J. Choi, C.-Y. Cher, H. Franke, H. Hamann, A. Weger, and P. Bose. Thermal-aware task scheduling at the system software level. In *Proc. Intl. Sym. on Low power electronics and design (ISLPED)*, pages 213–218, 2007.

[12] S. H. Choi, B. C. Paul, and K. Roy. Novel sizing algorithm for yield improvement under process variation in nanometer technology. In *Design Autoion Conf. (DAC)*, 2004.

[13] A. Datta et al. Statistical modeling of pipeline delay and design of pipeline under process variation to enhance yield in sub-100nm technologies. In *DATE*, 2005.

[14] P. Dutta, M. Grimmer, A. Arora, S. Bibyk, and D. Culler. Design of a wireless sensor network platform for detecting rare, random, and ephemeral events. In *Proc. Sym. on Information processing in sensor networks (IPSN)*, 2005.

[15] S. Garg and D. Marculescu. On the impact of manufacturing process variations on the lifetime of sensor networks. In *CODES/ISSS*, 2007.

[16] S. Ghosh, S. Bhunia, and K. Roy. A new paradigm for low-power, variation-tolerant circuit synthesis using critical path isolation. In *Intl. Conf. on Computer-aided design*, 2006.

[17] A. Golda and A. Kos. Temperature influence on energy losses in mosfet capacitors. *Microelectronics and Reliability*, 44(7):1115–1121, 2004.

[18] Justin Gregg and Tom W. Chen. Post silicon power/performance optimization in the presence of process variations using individual well-adaptive body biasing. *T. VLSI Syst.*, 15(3), 2007.

[19] ASU NIMO Group. PTM Hspice models. http://ptm.asu.edu.

[20] T. He et al. Achieving Real-time Target Tracking Using Wireless Sensor Networks. *IEEE RTAS*, 2006.

[21] H. Huang, G. Quan, and J. Fan. Leakage temperature dependency modeling in system level analysis. In *Proc. Intl. Sym. on Quality Electronic Design (ISQED)*, pages 447–452, 2010.

[22] K. Jeong, A.B. Kahng, and K. Samadi. Impact of Guardband Reduction On Design Outcomes: A Quant. Approach. *IEEE Trans. on Semiconductor Manufacturing*, 22(4):552–565, 2009.

[23] K. Kang, B. C. Paul, and K. Roy. Statistical timing analysis using levelized covariance propagation considering systematic and random variations of process parameters. *ACM Trans. Des. Autom. Electron. Syst.*, 11(4):848–879, 2006.

[24] Aman Kansal, Jason Hsu, Sadaf Zahedi, and Mani B. Srivastava. Power management in energy harvesting sensor networks. *ACM Trans. Embed. Comput. Syst.*, 6, 9 2007.

[25] V. Khandelwal and A. Srivastava. Variability-driven formulation for simultaneous gate sizing and post-silicon tunability allocation. In *Intl. Sym. on Physical Design (ISPD)*, 2007.

[26] A. Lachenmann, P. Marrón, D. Minder, and K. Rothermel. Meeting lifetime goals with energy levels. In *ACM SenSys*, 2007.

[27] P. Levis et al. TinyOS: An operating system for sensor networks. *Ambient Intelligence*, pages 115–148, 2005.

[28] J. W. S. Liu, W.-K. Shih, K.-J. Lin, R. Bettati, and J.-Y. Chung. Imprecise computations. *Proc. of the IEEE*, 82(1):83–94, 1994.

[29] Mateusz Malinowski et al. CargoNet: a low-cost micropower sensor node exploiting quasi-passive wakeup for adaptive asychronous monitoring of exceptional events. In *SenSys*, 2007.

[30] T. Matsuda et al. A power-variation model for sensor node and the impact against life time of wireless sensor networks. In *ICCE*, 2006.

[31] D. McIntire et al. The low power energy aware processing (LEAP) embedded networked sensor system. In *IPSN*, pages 449–457, 2006.

[32] O. Neiroukh and X. Song. Improving the process-variation tolerance of digital circuits using gate sizing and statistical techniques. In *DATE*, 2005.

[33] A. Pant, P. Gupta, and M. van der Schaar. Software adaptation in quality sensitive applications to deal with hardware variability. In *IEEE Great Lakes Sym. on VLSI*, pages 85–90, 2010.

[34] R. Puri, L. Stok, and S. Bhattacharya. Keeping hot chips cool. In *Proc. Design Automation Conf. (DAC)*, pages 285–288, 2005.

[35] V. Raghunathan, S. Ganeriwal, and M. Srivastava. Emerging techniques for long lived wireless sensor networks. *IEEE Communications Magazine*, 44(4):108–114, 2006.

[36] V. J. Reddi, M. S. Gupta, M. D. Smith, G. Wei, D. Brooks, and S. Campanoni. Software-assisted hardware reliability: abstracting circuit-level challenges to the software stack. In *Proc. Design Automation Conf. (DAC)*, pages 788–793, 2009.

[37] S. M. Rumble, R. Stutsman, P. Levis, D. Mazières, and N. Zeldovich. Apprehending joule thieves with Cinder. In *MobiHeld*, 2009.

[38] T. Sakurai and A.R. Newton. Alpha-power law mosfet model and its applications to cmos inverter delay and other formulas. *IEEE Journal of Solid-State Circuits*, 25(2):584 –594, 4 1990.

[39] J. Tschanz et al. Adaptive body bias for reducing impacts of die-to-die and within-die parameter variations on microprocessor frequency and leakage. In *ISSCC*, 2002.

[40] U.S. Climate Reference Network (USCRN). Hourly temperature data. www.ncdc.noaa.gov/crn/, 2010.

[41] L. Wanner, C. Apte, R. Balani, P. Gupta, and M. Srivastava. A case for opportunistic embedded sensing in presence of hardware power variability. In *HotPower*, 2010.

[42] L. Wanner, R. Balani, S. Sahedi, C. Apte, P. Gupta, and M. Srivastava. Variability-aware duty cycle scheduling in long running embedded sensing systems. In *DATE*, 2011.

[43] G. Wiedenhoft, L. Wanner, G. Gracioli, and A. Fröhlich. Power management in the EPOS system. *Oper. Syst. Rev.*, 42(6), 2008.

[44] S. Zahedi, M.B. Srivastava, C. Bisdikian, and L.M. Kaplan. Quality Tradeoffs in Object Tracking with Duty-Cycled Sensor Networks. In *Real-Time Systems Symp. (RTSS)*, 2010.

[45] H. Zeng, C. S. Ellis, Alvin R. L., and A. Vahdat. ECOSystem: managing energy as a first class operating system resource. *SIGOPS Oper. Syst. Rev.*, 36(5), 2002.