Communication Architecture Based Power Management for Battery Efficient System Design *

Kanishka Lahiri
Dept. of ECE
UC San Diego
klahiri@ece.ucsd.edu

Anand Raghunathan
C&C Research Labs
NEC USA
anand@nec-lab.com

Sujit Dey
Dept. of ECE
UC San Diego
dey@ece.ucsd.edu

ABSTRACT

Communication-based power management (CBPM) is a new battery-driven system-level power management methodology in which the system-level communication architecture regulates the execution of various system components, with the aim of improving battery efficiency, and hence, battery life. Unlike conventional power management policies (that attempt to efficiently shut down idle components), CBPM may delay the execution of selected system components even when they are active, in order to adapt the system-level current discharge profile to suit the battery's characteristics.

In this paper, we present a methodology for the design of CBPM based systems, which consists of system-level performance and power profiling, battery discharge analysis, instrumentation of system components to facilitate CBPM, definition of CBPM policies, and generation of the CBPM-based system architecture. We present extensive evaluations of CBPM, and demonstrate its application to the design of an IEEE 801.11 Wireless LAN MAC processor system. Our results indicate that CBPM based systems are significantly more battery efficient than those based on conventional power management techniques. Further, we demonstrate that CBPM enables design-time as well as run-time tradeoffs between system performance and battery life.

Categories and Subject Descriptors

C.5.4 [**Computer System Implementation**]: VLSI Systems

General Terms

Algorithms, Design, Experimentation

Keywords

Embedded Systems, Communication Architectures, Battery Efficiency, Power Management, Low Power Design

1. INTRODUCTION

Improvements in wireless communication and semiconductor fabrication technologies are key factors driving a rapidly expanding market for battery powered electronics. Unfortunately, projections of the complexity, functionality and performance of these electronic systems far exceed projected improvements in battery technologies, leading to a widening "battery gap" [1, 2, 3]. Bridging this gap is a challenge that system designers must face for the foreseeable future.

The need to improve battery life has, in large part, driven research and development of low power design techniques for electronic circuits and systems [4, 5, 6]. Low power design successfully reduces the energy that is drawn from the battery, and hence improves battery life to some extent. However, truly maximizing battery life requires an understanding of both the source of energy and the system that consumes it. It has been shown that the amount of energy that can be supplied by a battery varies significantly, depending on the manner in which the energy is drawn [3],[7]-[12]. Consequently, there is a need for "battery efficient", (not merely energy efficient) design techniques that are specifically tailored towards the characteristics and limitations imposed by battery-powered systems.

In this paper, we present *Communication Based Power Management* (CBPM), a new *battery driven* design methodology that aims at improving battery lifetime beyond what is achievable using state-of-the-art techniques for low power design. CBPM is implemented in the system-level communication architecture, and utilizes the communication architecture's ability to regulate the time profile of the system's power consumption.

1.1 Paper Overview and Contributions

Communication based power management (CBPM) is a system-level power management methodology that tailors the run time system power profile to ensure efficient battery discharge, in order to maximize battery life. CBPM policies exercise dynamic, regulatory control over the system components, by proactively blocking execution of selected components to prevent inefficient use of the battery, while minimally impacting performance. In this paper, we present a methodology and algorithms for the design of CBPM based systems. We illustrate the benefits of CBPM by applying it to the design of an IEEE 802.11 MAC processor system. We present results that indicate that CBPM results in a degree of battery efficiency superior to those obtained using traditional power management approaches. We also demonstrate that CBPM policies can be configured to allow for flexible and fine-grained trade-offs between battery life and performance.

1.2 Related Work

Recent work in the area of battery efficient design includes modeling techniques to efficiently and accurately capture important electro-chemical effects during battery discharge (*e.g.,* [8]-[11]. These models estimate battery lifetime and capacity under a variety of discharge scenarios, and are a key part of a battery driven design environment. Techniques for designing battery-efficient systems include battery driven speed and voltage setting [12, 13]. These approaches are typically aimed at general purpose computing systems, and hence do not exploit characteristics of the application. Battery driven voltage setting and task scheduling for application specific systems are described in [7, 14]. While the latter can be applied only to systems for which a static schedule exists, neither approach lends itself to dynamic control over the system power profile. Extensions to conventional dynamic power management techniques have been suggested in [15], to take battery constraints into account. Their approach is a reactive one, based on monitoring the remaining charge in the battery. Battery scheduling and management for multi-battery systems ([15]-[18]) do not address system power consumption, but optimize the battery subsystem to best satisfy power requirements. The Smart Battery System [19], an emerging industry standard, aims at standardizing communication between batteries and the systems they power, to facilitate interoperability and improved battery efficiency.

To our knowledge, ours is the first contribution to battery efficient design that provides techniques to dynamically regulate the system power profile, by exploiting the system-level communication architecture.

1.3 Paper Organization

In the next section, we provide background on batteries and system-level communication architectures. In Section 3, we present basic concepts of CBPM, and illustrate its benefits through an example. In Section 4, we present a methodology for designing CBPM based systems. Finally, in Section 5, we present results of experiments conducted while applying CBPM to the design of an IEEE 802.11 MAC processor.

* Supported by the National Science Foundation, grant 99-12414

DAC 2002, June 10-14, 2002, New Orleans, Louisiana, USA.

2. BACKGROUND

In this section, we describe battery characteristics that are taken into account by the CBPM methodology. We also describe relevant concepts associated with system-level communication architectures.

2.1 Battery Background

Terminology: The *standard capacity* of a battery is the energy that can be extracted from a battery when discharged under standard load conditions. For example, a typical lithium ion battery may have standard capacity 500 *mAh* when discharged at a constant current of 125 *mA*, at $25^{\circ}C$. The *actual capacity* of a battery is the amount of energy that the battery delivers under a given load, and is usually used (along with battery life) as a metric to judge the battery efficiency. While the actual capacity may differ from the standard capacity (depending on the load conditions), it cannot exceed a certain *theoretical capacity*. In this paper, *battery capacity* is used to refer to the actual capacity of the battery.

Rate Capacity Effects: Several electro-chemical effects make battery capacity sensitive to the discharge current profile. Significant among them are *rate capacity effects*, which result in a dependency between battery capacity and the magnitude of the discharge current [11].[1] Battery life depends largely on the availability of active reaction sites throughout the cathode. During discharge, chemical reactions cause the creation of inactive cathode sites, eventually leading to a state of discharge. During time intervals when the current drawn is small, inactive reaction sites are uniformly distributed throughout the material of the cathode. However, during intervals when the discharge current is larger, the outer surface of the cathode gets covered with inactive sites, making many internal active sites unreachable, effectively decreasing battery capacity. This rate capacity effect contributes to a significant dependence of battery capacity on the system's power profile [11].

2.2 System-level Communication Architectures

System-level communication architectures serve as a fabric to integrate various system components, and provides a medium for interacting system components to communicate. The *topology* of communication architectures can range from a single channel (shared bus), to a network of channels interconnected by *bridges*. System components include those that can initiate a communication transaction, called *bus masters* (*e.g.*, CPUs, DSPs, *etc.*), and those that merely respond to transactions initiated by a master, called *slaves* (*e.g.*, memories). Since channels are often shared by several masters, *communication protocols* include arbitration mechanisms to ensure that exactly one master has control of the channel at any given time. Typical arbitration mechanisms include *round-robin* access, *priority based* selection, *etc.*, which are implemented in centralized or distributed *arbiters*. The communication protocol also defines other communication functions, such as handshaking, burst transfers, and split transactions.

3. COMMUNICATION BASED POWER MANAGEMENT

In this section, we first explain the basic concept of communication based power management (CBPM), and then illustrate, through an example, the impact of CBPM on battery efficiency.

3.1 Basic Concept

Communication based power management makes use of the system-level communication architecture to regulate the execution of system components in order to obtain desirable system-level power profiles and improved battery efficiency. CBPM functionality differs significantly from conventional dynamic power management (DPM) techniques, which only aim to prevent wastage of energy during periods when the system or its components are idle. CBPM also differs from techniques that merely target an overall reduction in average power consumption. CBPM exercises dynamic and proactive control over system components, delaying their execution even if they are not in an idle state. CBPM based designs include mechanisms to block the execution of certain system components and force them to idle. Each component in a CBPM based system is internally power managed so as consume significantly less power when blocked than when active. By dynamically blocking the execution of certain components, CBPM can regulate the power profile of the system.

[1] Though CBPM does not explicitly target other effects, such as *charge recovery*, the battery life improvements that we report are generated using battery models that do account for these effects.

CBPM Rationale: While the aim of proactively controlling the execution of components in order to regulate the system power profile could be realized in many ways, we chose to use the system-level communication architecture for the following reasons:

Minimal hardware overhead: The communication architecture is a fabric that physically and logically integrates all the system components together. Embedding power management functionality in the communication architecture eliminates the need to include a separate power management unit and connect each component to it.

System-level visibility: To exercise control over the system power profile, power management policies need access to system wide information (*e.g.*, current execution states of different system components). By virtue of its connectivity, the communication architecture lends itself naturally to (i) effectively monitoring and detecting changes in the execution states of system components, and (ii) providing an efficient delivery mechanism for power management decisions to any system component.

Significant impact on power profile: The communication architecture directly controls the timing of system-level communications, and hence the execution of system components. In particular, the communication architecture can be used to control the timing of component idle and active states, impacting the component's power profile. Since the system power profile is the sum of the component power profiles, the communication architecture can be used to significantly impact the system power profile, as shown in the next subsection.

In summary, due to its system-level connectivity, and its potential to control the power profile of the system, the communication architecture naturally lends itself to use in a battery efficient system design methodology.

3.2 Example: IEEE 802.11 MAC Processor

We next illustrate the impact of CBPM by considering its application to an IEEE 802.11 MAC processor [20]. We compare the execution of the original performance optimized MAC processor with a CBPM based MAC processor. Note that, the original MAC processor incorporates DPM to minimize power consumed by idle components. For each design, we analyze representative power profiles, and battery efficiency, as the system processes a sequence of MAC frames.

The details of the MAC processor are presented in Section 5. However, for the sake of our illustration in this section, it is sufficient to consider only the system power profiles. Figure 1(a) shows a snapshot of the current discharge profile of the MAC processor as it processes 3 MAC frames.[2] The first is of length 54 bytes, and the next two (which arrive back to back) are of length 1414 bytes each. This snapshot is part of a longer profile obtained by simulating the MAC processor as it processes frame arrivals during 1.5 minutes of streaming video over a wireless LAN.

Using a stochastic model of a Lithium-ion battery (250 *mAh* capacity, rated discharge current of 125 *mA* [9]), we measured battery life and ca-

[2] Current discharge profiles are obtained by dividing the power profiles by the supply voltage (assumed to be constant).

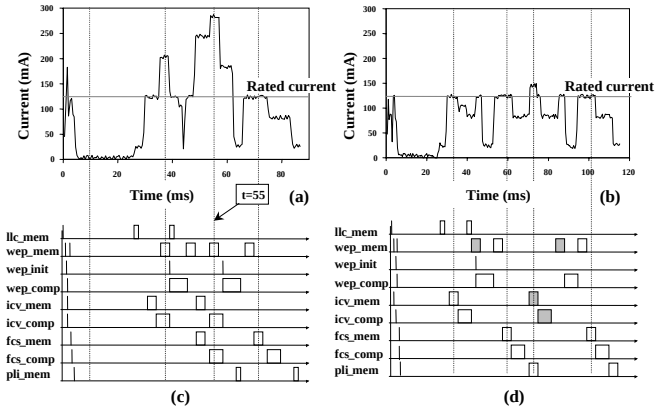

Figure 1: (a) Discharge profile without CPBM (b) Discharge profile with CBPM (c,d) Symbolic execution traces identifying component macro-states

pacity under the given profile. Under these conditions of discharge, the battery capacity and lifetime obtained are 154.3 *mAh* and 3209.2 *sec* respectively. We observe from Figure 1(a) that the current profile frequently exhibits regions where the current drawn is significantly higher than the rated discharge current of 125 *mA*.

To understand how CBPM improves battery efficiency, consider a symbolic execution trace of the system. An example of such a trace is shown in Figure 1(c), corresponding to the current profile of Figure 1(a). The trace depicts 9 "macro-states", which indicate the specific sub-tasks that each of 5 components (`LLC, WEP, ICV, FCS, PLI`) perform at any given time. For example, `WEP`, which is the component responsible for encrypting frame data, may be in one of several macro-states depending on whether it is reading a frame body from memory (`wep_mem`), initializing the key state array (`wep_init`), or computing an encrypted frame body (`wep_comp`).

The state of the system at any point of time is determined by examining the macro-states of individual components. From Figures 1(a) and (c), it is clear that different system states exhibit wide variation in the current drawn, depending on the individual macro-states of the system components. For example, Figure 1(c) indicates that, at $t = 55$ *ms*, the system state is one where two HW components (`ICV, FCS`) are engaged in lengthy iterative computations (`icv_comp, fcs_comp`), while a third (`WEP`) is accessing memory over a shared bus (`wep_mem`). Figure 1(a) indicates that this system state leads to a drawn current of 280 *mA*.

We next consider the processing of the same sequence of MAC frames, now with the CBPM policies enabled. A snapshot of the new current profile is shown in Figure 1(b), while the corresponding symbolic execution trace is shown in Figure 1(d). By comparing the symbolic execution traces in Figures 1(c) and (d), it can be seen that the CBPM policy effectively delays the execution of certain macro-states (indicated by shading), which would otherwise have led to a violation of the rated discharge current. For example, Figure 1(d) shows that CBPM delays the occurrence of macro-state `wep_mem` to $t = 42$ *ms* so as to avoid overlap with the macro-state `icv_comp`. From the current discharge profile in Figure 1(a), it is clear that the CBPM policy greatly reduces the occurrence of system states that violate the rated current. Battery life of the new system was measured to be 10199 *sec* (a 3.2X improvement) and more importantly, battery capacity was measured to be 225.4 *mAh* (a 1.5X improvement). This example illustrates the significant impact CBPM can have on the system power profiles, and hence overall battery efficiency.

4. CBPM DESIGN METHODOLOGY

In this section, we first present a structured methodology highlighting the various steps involved in designing CBPM based systems. Next, we present a detailed description of each step, including the various algorithms and tools used.

4.1 CBPM Methodology: Overview

The inputs to our methodology (shown in Figure 2) are (i) a simulatable system specification mapped to a system architecture consisting of HW/SW components (ii) a system-level communication architecture to integrate the various components, (iii) typical environment stimuli or traces, (iv) performance and battery life objectives. The output of the methodology is a CBPM policy that is capable of dynamically regulating the system power consumption profile so as to ensure efficient battery discharge. From a hardware perspective, the result is a system with a communication architecture enhanced with new battery aware communication protocols.

In **Step 1**, HW/SW co-simulation is performed to generate execution traces and system power profiles under provided input stimuli. In **Step 2**, the specification mapped to each system component is decomposed into a set of macro-states, whose execution is under the control of the communication architecture. In **Step 3**, power profiling tools are used to estimate the average power consumption of each macro-state. In **Step 4**, performance analysis is used to prioritize the macro-states in terms of their performance criticalities. In **Step 5**, the CBPM policy is defined. In **Step 6**, system-level performance analysis and battery life estimation are performed, accounting for the defined CBPM policy. Incorporation of CBPM policies effectively change the timing of certain macro-states. This may lead to a change in the relative criticalities of the macro-states, requiring recalculation of macro-state criticalities each time the CBPM policy is modified. Hence, if battery life and performance objectives are not met, **Steps 4, 5, 6** are repeated in an iterative manner. Finally, **Step 7** is executed, where the CBPM policy is implemented in the system-level communication architecture.

4.2 CBPM Methodology: Details

In this sub-section, we consider, in turn, the various steps outlined above.

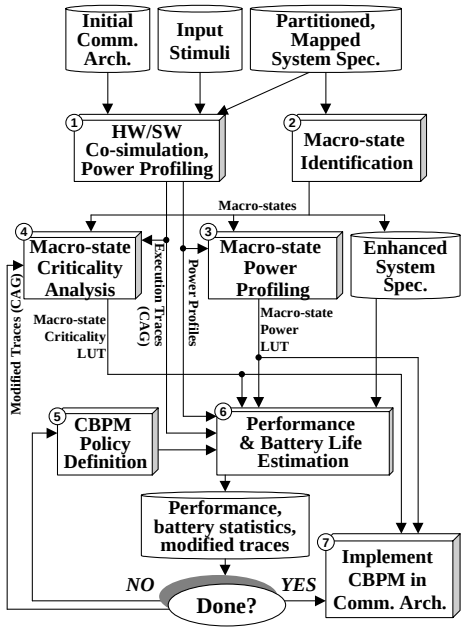

Figure 2: Communication Based Power Management (CBPM) design methodology

Step 1 — HW/SW co-simulation, trace generation: HW/SW co-simulation and power co-estimation of the entire system is performed, under provided input stimuli. The resulting execution traces and power profiles are used in **Steps 3, 4** and **6**. In our implementation, we used the framework described in [21].

Step 2 — Macro-state identification: A component's impact on system performance, and its contribution to the system power profile may exhibit significant variation, depending on which part of its specification it is currently executing. To take this dependence into account, we introduce the concept of a component macro-state. For each component C, we define a set of macro-states $s_0, \ldots, s_{n_C}$, and modify the specification of the component to make these macro-states identifiable at run-time. The procedure for identifying macro-states is as follows:

2A. For each component, label each basic block in the Control Flow Graph (CFG) of its specification as a macro-state s_i.

2B. Use performance analysis (**Step 4**) to measure $delay(s_i)$, the execution time of each macro-state s_i under provided inputs.

2C. The macro-states so obtained must satisfy:

- **Battery Constraint:** $k_1\tau < delay(s_i) < k_2\tau, i = 1, \ldots, n_C$, where τ is the battery time constant, n_C is the number of macro-states of component C, $k_1 < 1$ and $k_2 > 1$ are positive constants. This constraint provides a means to control the granularity at which CBPM regulates the power profile.[3]
- **Complexity Constraint:** $n_C < N$, where N is a constant. This constraint can be used to control the run-time overhead of keeping track of component macro-states.

If these conditions are satisfied, **Step 2E** is executed. Otherwise, **Step 2D**, (a clustering step), and **Steps 2B, 2C** are repeated.

2D. For each s_i for which $delay(s_i) < k_1\tau$, an attempt is made to combine s_i with neighboring macro-states. This is performed in a greedy manner, where s_i is combined with its neighbor t_i that has maximum delay. Due to control flow requirements, the clustering of s_i with t_i may force expansion of the cluster to include other macro-states.

2E. For each macro-state s_i, a blocking communication event e is identified such that s_i is dependent on e (unless e executes, s_i cannot execute). If no such events can be found, an explicit blocking communication is added at the beginning of s_i. Additionally, these communication events are modified to carry a macro-state identifier.

[3] The benefits of fine grained control over the power profile are limited by (typically large) battery time constants.

The output of **Step 2** is an enhanced specification for each component, containing specific communications that carry information about the current macro-state of the component.

Step 3 — Macro-state Power Profiling: The average power consumption of each macro-state (identified in **Step 2**) is estimated as follows:

- **3A.** From the system execution traces, construct symbolic traces for each component, which indicate timing of each macro-state;
- **3B.** From the power profiles and symbolic traces, calculate $pow(s_{ij})$, the average power consumed by the j^{th} instance of s_i. [4]
- **3C.** Calculate $pow(s_i)$, the average power consumed by macro-state s_i, by averaging over all instances of s_i.

The output of **Step 3** is a *macro-state power table*, consisting of tuples, $\langle s_i, pow(s_i)\rangle$ representing macro-states, and their respective average power consumptions.

Step 4 — Macro-state Criticality Analysis: The sensitivity of overall system performance to the execution times of each macro-state is estimated using a trace based system-level performance analysis framework [22]. The performance analysis proceeds in two phases. In a pre-processing phases, execution traces (obtained in **Step 1**) are analyzed to construct a Communication Analysis Graph (CAG), an abstract yet accurate representation of the system execution traces, consisting of time stamped vertices that represent computations and communications, and edges that represent control flow dependencies. In the second phase, the tool provides estimates of system performance for any specified communication architecture by rapidly transforming the CAG, to take into account effects of the communication architecture (*e.g.*, arbitration, burst transfers, CBPM policies, topology *etc*). The output of the tool is a transformed CAG (Figure 3) that provides useful performance statistics for any specified communication architecture. Next, the CAG is analyzed to determine the criticalities of each macro-state. Note that, when **Step 4** is executed for the first time, the input is a CAG which represents system execution under the initial communication architecture. For subsequent executions, the input CAG represents system execution under a CBPM enhanced communication architecture (Figure 2).

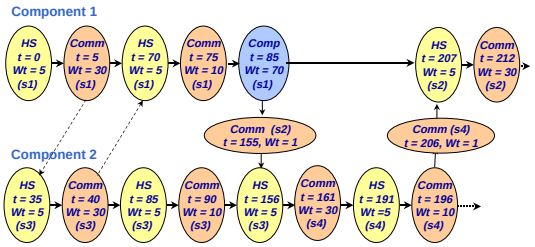

Figure 3: Modified communication analysis graph (CAG) that reflects effects of a priority based shared bus

- **4A.** Each vertex in the CAG is labeled with the macro-state it is associated with (Figure 3).
- **4B.** For each occurrence s_{ij} of a macro-state s_i, the start time of the first vertex in s_i is delayed by δ.
- **4C.** The CAG is analyzed[22] to estimate the effect of this delay on system performance, $crit(s_{ij})$. The results of this analysis are the values of $crit(s_{ij})$ for all i, j.
- **4D.** The average criticality of macro-state s_i, $crit(s_i)$ is obtained by averaging $crit(s_{ij})$ over all j.

The output of **Step 4** is a *macro-state criticality table*, which consists of tuples, $\langle s_i, crit(s_i)\rangle$, representing macro-states, and their respective criticalities.

Step 5 — CBPM Policy Definition: The two aims while defining the CBPM policy are (i) minimize violations of battery rate capacity constraints, and (ii) minimize the extent to which the execution of performance critical macro-states are blocked. The basic CBPM policy is as follows. Let C = set of currently executing macro-states. Let S = set of pending macro-states (those awaiting a grant from CBPM). At each CBPM arbitration, select a set of macro-states $S' \subseteq S$ such that the following conditions hold:

$$\sum_{x \in S' \cup C} pow(x) < P_{max} \tag{1}$$

$$crit(x) > crit(y), \forall (x,y) \mid x \in S', y \in S - S' \tag{2}$$

Clearly the power constraint P_{max} in Eqn 1 is a critical determinant of the extent to which performance is traded off for battery life improvements. A larger value of P_{max} leads to more macro-states being included in S', resulting in fewer blocked components, enhanced performance, and decreased battery efficiency. A smaller value of P_{max} constrains the size of S, inhibiting the number of simultaneously active components, potentially trading battery efficiency for performance. Eqn 2 ensures that if all the pending macro-states cannot be granted for a given value of P_{max}, the policy grants the most performance critical subset from the set of pending macro-states.

A natural question that arises when blocking the execution of system components, is that of deadlock [23]. To see that CBPM policies cannot result in deadlock, note that, in a CBPM based system, communications are blocking, and components are internally power managed. Hence, when components are blocked, they consume very little power, and hence, free up the power "resource" for other components to use. This prevents circular wait, a necessary condition for deadlock.

Step 6 — Performance and Battery Life Estimation: Detailed analysis is carried out in order to examine the impact of the CBPM policy on system performance and battery life. For performance analysis, we use the technique described in **Step 4**, which also generates a CAG with effects of the CBPM policies incorporated. The battery efficiency of the system is estimated as follows:

- **6A.** The CAG derived via performance analysis is used to generate a symbolic execution trace of the system indicating the timing of various macro-states, considering the effects of the new communication protocols.
- **6B.** A new system power profile is generated from the symbolic execution traces and the macro-state power table.
- **6C.** Battery discharge is simulated using the power profiles to obtained battery capacity and lifetime estimates. In our work, we used a stochastic model of a lithium ion battery of 250 *mAh* capacity, with rated discharge current 125 *mA* [9].

If desired performance and battery goals are met, **Step 7** is executed. Otherwise, the CBPM policy is modified by adjusting the power constraint P_{max}, and **Steps 4, 5, 6** are executed iteratively. Note that, in these iterations, macro-state criticality analysis (**Step 4**) is driven by a CAG that includes effects of the CBPM policy.

Step 7 — CBPM Implementation: In this step, the CBPM policy is implemented in the communication architecture. Figure 4 shows a shared bus based communication architecture enhanced to implement CBPM. Requests from the components to the bus arbiter optionally carry a macro-state identifier. Requests to the arbiter are of two types (i) Type I requests: these

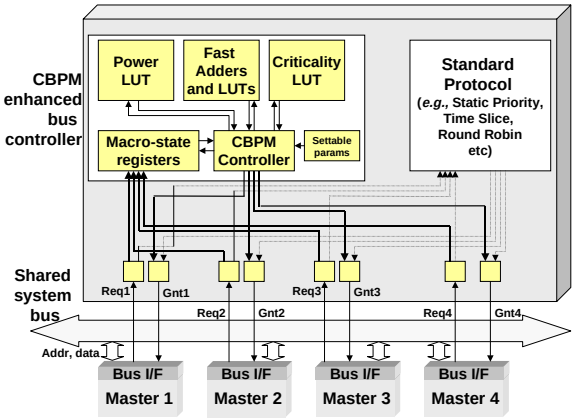

Figure 4: CBPM based communication architecture

[4] Note that, this estimate includes the power consumed by all the components associated with a macro-state s_i. For example, when s_i represents a large volume data transfer, the estimate includes the power consumed by the master and slave components, bus interface logic, and bus lines.

occur at macro-state boundaries, and are requests to transition to the next macro-state, and (ii) Type II requests: these represent conventional bus access requests. Type I requests (which occur at macro-state time-scales), are handled by the CBPM circuitry. Type II requests occur at small time-scales (potentially every bus cycle) and are handled by the underlying bus protocol (*e.g.*, priorities, TDMA, *etc*). Multiplexers are used to route requests to the appropriate logic in the arbiter. The CBPM circuitry includes the following:

- Macro-state registers that store a current and pending macro-state identifier for each component;
- Power LUT that stores macro-state power table;
- Criticality LUT that stores the macro-state criticality table;
- CBPM controller, which implements the arbitration algorithm for Type I requests. The algorithm is based on the policy defined in **Step 5**. At each arbitration, the total power consumed by the currently executing macro-states is calculated. This is obtained either by (i) using fast adders to add power numbers obtained by looking up the Power LUT, or (ii) using LUTs which store precomputed results for commonly occurring macro-state combinations. In parallel, the set of pending macro-states are sorted into descending order of criticality by looking up the Criticality LUT. Starting with the most critical, the controller grants pending states till either (i) there are no more pending macro-states, or (ii) granting another macro-state violates the power constraint.
- Settable parameters are stored in a register to provide for run-time configuration of the CBPM policy. For example, the power constraint P_{max} can easily be made run-time settable, permitting dynamic flexibility in trading off performance for battery life.

5. EXPERIMENTS

In this section, we describe (i) the IEEE 802.11 MAC processor system used for our experiments (full details of our implementation are available in [24]), (ii) the experimental methodology, and (iii) experimental results, including battery versus performance trade-offs, comparison with conventional power management techniques, and HW implementation.

5.1 IEEE 802.11 MAC Processor System

HW/SW Architecture: The MAC processor architecture (Figure 5(b)) consists of two embedded SPARCLite CPUs [25], 3 HW co-processors, 2 memories, and 3 queue buffers. The component `llc_c` is a HW implementation of the Logical Link Control Interface (LLC) task. It receives frames from the LLC layer, writes them in `mem_1`, and enqueues frame addresses in `Queue1`. Addresses are read from the queue by (i) `wep_c`, a HW co-processor implementing the Wired Equivalent Privacy (WEP) task, which encrypts frame data, and (ii) `sparc_1`, which computes an Integrity Checksum Vector (ICV) for each frame. The ICV and WEP tasks proceed in parallel. Encrypted frames are written to `mem_2`, and frame addresses to `Queue2`. The embedded CPU `sparc_2` implements tasks HDR, which generates the MAC header, FCS, which checksums the entire frame, and MAC_CTRL, which implements CSMA/CA, the channel access algorithm [20]. Once processing is complete and the frame can be transmitted, `pli_c`, a HW implementation of the Physical Layer Interface (PLI) dequeues frame addresses from `Queue3`, retrieves frames from `mem_2`, and passes them to the physical layer hardware.

System-level Communication Architecture: The communication architecture consists of dedicated channels for low volume communications (*e.g.*, synchronization between `sparc_1` and `wep_c`), and two 32-bit shared busses for data transfers between system components. The busses are interconnected by a `bridge`, and a static priority based protocol [26] is employed on each bus. Components `llc_c, wep_c, sparc_1`, and the `bridge` contend for access to `bus_1`, while components `bridge, sparc_2` and `pli_c` contend for access to `bus_2`. Note that, the CBPM methodology is general, and can be applied to systems with other communication architectures as well.

5.2 Experimental Methodology

The MAC processor was designed in the POLIS [27] environment, using a combination of Esterel and C to specify the system tasks. For performance analysis, we used a tool described in [22], while system power profiling was performed using a modified version of the power estimation framework described in [21]. On average, three macro-states were identified for each component. For example, the identified macro-states for `wep_c` were `wep_init` (state array initialization), `wep_mem` (block transfer of data from memory) and `wep_comp` (frame encryption). Other macro-states are listed in Figure 1(c,d). Battery life estimation was performed using a stochastic model of a Lithium-ion battery [9], with standard capacity 250 *mAh*, at a rated current of 125 *mA* (typical values for batteries commonly found in cell phones, PDAs *etc* [28]). Simulations were driven by actual 802.11 traffic, obtained by running ETHEREAL [29], a packet capture software on a laptop PC connected to a wireless LAN, under different applications. Battery efficiency was measured in terms of battery capacity and lifetime. Performance (maximum data rate) was estimated using the PTOLEMY simulation environment [30].

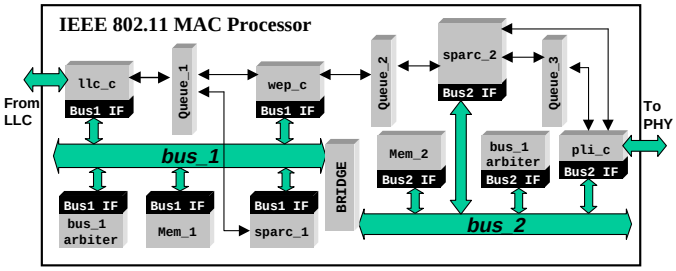

Figure 5: IEEE 802.11 MAC Processor Architecture

5.3 Battery Efficiency and Performance Trade-Off

To evaluate the effectiveness of the CBPM techniques, we experimented with different configurations of the MAC processor, each with a different setting of the power constraint. Varying the power constraint effectively changes the subset of macro-states that the CBPM policy chooses to grant. Figure 6 shows (i) battery capacity (*mAh*) as well as (ii) data rate (*Kbps*) of the MAC processor under different configurations of the CBPM policy. The trace consists of 4818 frames generated by 1.5 minutes of streaming video over an 802.11 based wireless network on a laptop running the Microsoft Windows Media Player [31]. From Figure 6 we can make the following observations:

- The CBPM based MAC processor shows a substantial increase in battery efficiency. Battery lifetime improvements from 3209.2 seconds to 10199.9 seconds are observed (3.2X), while more importantly, capacity improves from 154.3 *mAh* under configuration 0 (CBPM disabled) to 225.4 *mAh* under configuration 6 (1.5X).
- Improvements in battery capacity come at a price in terms of system performance. For example, configuration 6 provides for an improvement in battery lifetime of 3X, while degrading the data rate by 2.3X. However, as shown in the next experiment, the performance/battery trade off resulting from CBPM is significantly superior to what can be achieved using conventional power management techniques.

5.4 Comparison with other Power Management Techniques

For this experiment, we consider a design of the MAC processor that supports clock frequency scaling, a commonly used power management

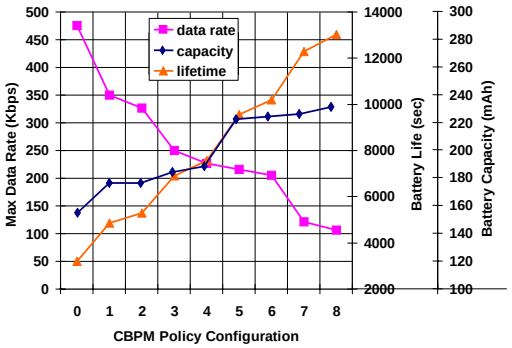

Figure 6: Battery efficiency *vs* performance for different configurations of CBPM policy

technique for extension of battery life in mobile appliances [12], the objective being to compare the trade-off characteristics obtained via frequency scaling versus those obtained via CBPM. We first operated the MAC processor over a wide performance range, scaling the clock frequency of each component by a constant factor over a discrete range of values. For each frequency, we exercised the MAC processor with a long trace of frame arrivals, and evaluated battery lifetimes and battery capacity. Next, we operated the CBPM based MAC processor over same performance range and measured battery capacity for each point. The two trade-off curves thus obtained are shown in Figure 7, based on which we make the following observations:

- CBPM is superior to frequency scaling as a tool for trading off battery efficiency for performance, because (i) it incurs a smaller performance penalty than frequency scaling for equivalent improvements in capacity, and (ii) it provides more battery-efficient solutions for equivalent performance. For example, to improve the battery capacity from 128 *mAh* to 182 *mAh* (an improvement of 42%), frequency scaling incurs a performance degradation of 22%, while that of the CBPM based system is only 7%.
- A major problem with the frequency scaling approach in improving battery efficiency is observed. In some cases, as the frequency is reduced, (decreasing performance), battery capacity actually decreases. This occurs because, decreasing the clock frequency results in the overlapping of various power hungry macro-states (which were disjoint at higher frequencies), leading to rated current violations. No such anomalies are observed with the CBPM trade-off curve.

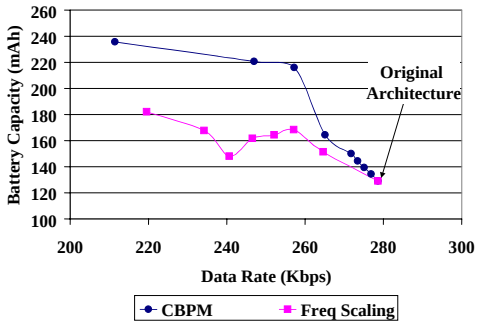

Figure 7: Comparison of the trade-off characteristics of the CBPM policy with frequency scaling

The experiment highlights the benefit of using a battery driven power management policy rather than one that merely aims at reducing average power.

A common dynamic power management (DPM) technique is idle shut down, where idle components are forced into a low power state. Note that, our original architecture (without CBPM) already incorporates DPM. Hence, the reported battery efficiency of the CBPM based MAC processor is over and above what is achievable using DPM.

5.5 HW Complexity

The CBPM enhanced bus arbiter was synthesized and mapped to a commercial 0.18μ standard cell library [32]. The area of the CBPM enhanced arbiter was found to be $15014\mu^2$, a 56% increase over the area of the original arbiter. To investigate the impact of this at the system-level, we estimated the total chip area by using fast synthesis for the HW co-processors, and data sheets for the memory and embedded CPUs. The overhead at the system level due to CBPM enhancements to the communication architecture was found to be less than 0.5%.

The delay of the enhanced bus arbiter was measured to be $4.6ns$, implying that single cycle CBPM arbitrations can be performed for bus speeds upto $218Mhz$. Significantly higher bus speeds can be supported by supporting multi-cycle arbitration for the less frequent CBPM arbitrations.

6. CONCLUSIONS

In this paper, we presented CBPM, a new power management methodology that can be used to dynamically regulate the system power profile to ensure efficient battery discharge, and demonstrated its application to the design of a battery-efficient IEEE 802.11 MAC processor. Results of our experiments indicate that CBPM can be used to design systems that are significantly more battery efficient than those based on conventional power management.

7. REFERENCES

[1] S. H. Gunther, F. Binns, D. M. Carmean, and J. C. Hall, "Managing the Impact of Increasing Microprocessor Power Consumption." Intel Technology Journal, First Quarter, 2001, available online at http://developer.intel.com/technology/itj/

[2] I. Buchmann, "Batteries in a Portable World." http://www.cadex.com.

[3] K. Lahiri, A. Raghunathan, S. Dey, and D. Panigrahi, "Battery driven system design: a new frontier in low power design," in *Proc. ASP-DAC/Int. Conf. VLSI Design*, pp. 261–267, Jan. 2002.

[4] A. R. Chandrakasan and R. W. Brodersen, *Low Power Digital CMOS Design*. Kluwer Academic Publishers, Norwell, MA, 1995.

[5] J. Rabaey and M. Pedram (Editors), *Low Power Design Methodologies*. Kluwer Academic Publishers, Norwell, MA, 1996.

[6] L. Benini and G. De Micheli, *Dynamic Power Management: Design Techniques and CAD Tools*. Kluwer, 1998.

[7] M. Pedram and Q. Wu, "Design considerations for battery-powered electronics," in *Proc. Design Automation Conf.*, pp. 861–866, June 1999.

[8] D. Rakhmatov and S. B. K. Vrudhula, "Time to failure estimation for batteries in portable systems," in *Proc. Int. Symp. Low Power Electronics & Design*, pp. 88–91, Aug. 2001.

[9] D. Panigrahi, C. F. Chiasserini, S. Dey, R. R. Rao, A. Raghunathan, and K. Lahiri, "Battery life estimation for mobile embedded systems," in *Proc. Int. Conf. VLSI Design*, pp. 55–63, Jan. 2001.

[10] L. Benini *et.al.*, "A discrete-time battery model for high-level power estimation," in *Proc. Design Automation & Test Europe Conf.*, pp. 35–39, Mar. 2000.

[11] M. Doyle, T. F. Fuller, and J. S. Newman, "Modeling of galvanostatic charge and discharge of lithium/polymer/insertion cell," *J. Electrochem. Soc*, vol. 140, pp. 1526–1533, June 1993.

[12] 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.

[13] "Mobile Intel Pentium III Processor featuring Intel SpeedStep (TM) Technology Performance Brief." http://developer.intel.com/design/mobile/

[14] J. Luo and N. K. Jha, "Battery-aware static scheduling for distributed real-time embedded systems," in *Proc. Design Automation Conf.*, pp. 444–449, June 2001.

[15] L. Benini *et.al.*, "Battery-driven dynamic power management," *IEEE Design & Test of Computers*, vol. 18, pp. 53–60, Apr. 2001.

[16] L. Benini *et.al.*, "Extending lifetime of portable systems by battery scheduling," in *Proc. Design Automation & Test Europe Conf.*, pp. 197–201, Mar. 2001.

[17] C. F. Chiasserini and R. R. Rao, "Energy Efficient Battery Management," *IEEE J. on Selected Areas in Communications*, vol. 19, pp. 1235–1245, July 2001.

[18] Q. Wu, Q. Qiu, and M. Pedram, "An interleaved dual-battery power supply for battery powered electronics," in *Proc. Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 387–390, Jan. 2000.

[19] "Smart Battery System Implementers Forum (http://www.sbs-forum.org).".

[20] "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications." IEEE Computer Society LAN MAN Standards Committee, IEEE Std 802.11-1999 Edition.

[21] M. Lajolo, A. Raghunathan, S. Dey, and L. Lavagno, "Efficient Power Co-Estimation Techniques for System-on-Chip Design," in *Proc. Design Automation & Test Europe Conf.*, pp. 27–34, Mar. 2000.

[22] K.Lahiri, A.Raghunathan, and S.Dey, "System level performance analysis for designing on-chip communication architectures," *IEEE Trans. Computer-Aided Design*, vol. 20, pp. 768–783, June 2001.

[23] A. Silberschatz, P. Galvin, and G. Gagne, *Operating System Concepts*. Wiley and Sons, 2001.

[24] K. Lahiri, A. Raghunathan, and S. Dey, "Battery efficient architecture for an 802.11 MAC processor," in *Proc. IEEE Int. Conf. on Communications*, Apr. 2002.

[25] Fujitsu Microelectronics Inc., "Sparclite series mb86934 addendum." 1994.

[26] "Peripheral Interconnect Bus Architecture." http://www.omimo.be.

[27] F. Balarin *et.al.*, *Hardware-software Co-Design of Embedded Systems: The POLIS Approach*. Kluwer Academic Publishers, Norwell, MA, 1997.

[28] "Panasonic Lithium Ion Batteries: Individual Datasheet CGR 17500." http://www.panasonic.com/industrial/battery/oem/chem/lithion/

[29] "Ethereal 0.8.18." http://www.ethereal.com.

[30] J. Buck *et.al.*, "Ptolemy: A framework for simulating and prototyping heterogeneous systems," *International Journal on Computer Simulation, Special Issue on Simulation Software Management*, vol. 4, pp. 155–182, Apr. 1994.

[31] "Microsoft Windows Media Player." http://www.microsoft.com/windows/windowsmedia/

[32] "UMC 0.18 μ Copper Process Low-Voltage SAGE(TM) Standard Cell Library." http://www.umc.com/english/process/