System-Level Power-Aware Design Techniques in Real-Time Systems

Osman S. Unsal, *Member, IEEE,* and Israel Koren, *Fellow, IEEE*

***Abstract*— Power and energy consumption has recently become an important issue and consequently, power-aware techniques are being devised at all levels of system design; from the circuit and device level, to the architectural, compiler, operating system and networking layers. In this survey we concentrate on power-aware design techniques for real-time systems. While the main focus is on hard real-time, soft real-time systems are considered as well. We start with the motivation for focusing on these systems and provide a brief discussion on power and energy objectives. We then follow with a survey of current research on a layer by layer basis. We conclude with illustrative examples and open research challenges. This work provides an overview of power-aware techniques for the real-time system engineer as well as an up-to-date reference list for the researcher.**

***Index Terms*— Real-time systems, power-aware design, microarchitecture, compiler, operating system, network**

I. Prologue

As we get closer to the limits of scaling in CMOS circuits, power and heat dissipation issues are becoming ever more important. In recent years, the impact of pervasive computing and the internet have accelerated this trend. The applications for these domains are typically run on battery-powered embedded systems. The resultant constraints on the energy budget require design-for-power as well as design-for-performance at all layers of system design. Even when battery-based constraints do not exist, energy must often be conserved, in general-purpose as well as in real-time systems. Consider, for example, the following facts:

- According to Dataquest, worldwide total power dissipation of processors in PC's was 160 megawatts in 1992, growing to 9000 megawatts by 2001. In the same vein, a typical server farm with 25000 square feet of space and 8000 servers is estimated to consume 2 megawatts of power [1].
- A computer owner in Britain demonstrated the extent of the thermal and heat density issues by placing a dish of aluminum foil above the chip inside his PC and frying an egg for breakfast [2] (cooking time 11 minutes, experiment details and photos available at [3]).

Real-time systems are often severely energy-constrained. The need to continue meeting critical task deadlines while staying within power and energy constraints presents interesting engineering challenges.

Manuscript received January 20, 2002; revised November 18, 2002. This work was supported in part by NSF grants EIA-0102696 and CCR-0205212.

Osman S. Unsal was with Electrical and Computer Engineering Department, University of Massachusetts, Amherst, MA 01003. He is now with Intel Labs Barcelona, Spain.

Israel Koren is with Electrical and Computer Engineering Department, University of Massachusetts, Amherst, MA 01003, USA.

It is important to note the conceptual difference between power-aware and low-power systems. In low-power design, the main goal is minimization of power. On the other hand, a power-aware system is one in which meeting power and energy goals is a significant design consideration and in which the system modifies its behavior based on current power/energy availability. At this point it is worthwhile to further clarify what is meant by power-aware design and to correct some misconceptions associated with this term:

- **Power-aware design does not necessarily imply minimization of power or energy.** On the contrary, some power-aware design goals may increase power/energy consumption. Consider the case of design for decreasing peak power in a processor: one method to attain this goal would be to use schemes which would intentionally delay the issue of some instructions to smoothen the instruction issue distribution and thus decrease the peak consumed power. However, delaying some instructions could lead to the application being finished later than it otherwise would, therefore increasing the energy consumption. Thus, this scheme would be a power-aware, but not a low-power, design.
- **Decreasing average power does not imply decreasing maximum power.** Those goals are non-orthogonal subsets of low-power design. Average power dissipation is calculated over the entire run of the application. Being an average of the power consumption distribution histogram, average power is distinct from maximum power which is the peak value of the histogram. An optimization meant to decrease average power could even lead to an increase in the maximum power used.
- **Power and energy efficiency are separate design goals.** Energy is the integral of power consumption over a time period. A power-efficient design might very aggressively decrease the clock rate, but this might not be an energy-efficient design since the performance of the application might degrade to such a degree that actual energy consumption increases.
- **Power-constrained applications are distinct from energy-constrained ones.** An application running on a finite source of energy such as batteries is an energy-constrained application whereas an application running on an infinite source such as solar power is power-constrained. Maximum available power and the energy budget are different metrics in power-aware design.
- **Energy-constrained systems do not always target energy minimization.** If the charge that is drawn from a battery solely depends on the battery capacity (the total

energy budget) then energy minimization would be a valid design goal for a battery equipped system. However, studies of batteries have shown their behavior to be unlike ideal capacitors, i.e., the charge depends not only on the battery capacity but also on the rate of discharge [4], [5], [6], [7]. Therefore, for those energy constrained systems the goal is battery lifetime extension which is separate from energy minimization.

In this paper we survey research into power-aware real-time systems. In Section II, we present a concise tutorial on system-level power-aware design. We then follow with a survey on current power-aware research issues in real-time systems. The survey is partitioned into four sections, covering architectural, compiler, OS, and network layers. We consider both hard and soft real-time systems in each section. However, we will devote more attention to hard real-time systems since several surveys exist for soft real-time systems covering the multimedia [8] and embedded [9] domains.

A. System-Level Power-Aware Design for Real-Time Systems

We assume familiarity with common concepts in real-time systems; for detailed information, the reader is encouraged to consult [10], [11], [12]. System-level power-aware design in real-time systems is a relatively new research area. Low-power had become an important parameter at the higher layers of system design by the mid - 1990's. Most of the new system-level low-power techniques initially targeted general-purpose computing systems. However, it soon became apparent that real-time systems present unique challenges and opportunities for system-level low-power design as demonstrated next.

- Real-time systems are usually severely power-constrained. In particular, space borne and multimedia systems are typically battery-operated and therefore have a limited *energy budget*. Real-time systems are also relatively more time-constrained compared to general-purpose systems. Therefore, the challenge is to save power while satisfying temporal guarantees.
- Some real-time applications such as avionics, robotics and deep space missions require systems with small form factors, which in turn mandates low heat dissipation. Since heat is a byproduct of power dissipation, low-power system-design ensures a more reliable system by limiting the heat produced.
- Real-time systems are typically over-designed to ensure that the temporal deadline guarantees are still met even if all tasks take up their worst-case execution time (WCET) to finish. Since, in the average case, tasks do not run until their WCET, this is very energy inefficient. System-level techniques can decrease this power dissipation through the use of power-aware task scheduling algorithms while preserving the temporal guarantees.
- Real-time systems are designed to be fault-tolerant. Fault-tolerance ensures reliability through replication of software/hardware resources. However, brute replication in turn, causes high power dissipation. System-level low-power techniques manage replication resources judiciously to reduce the required power.

System-level power-aware research for real-time systems is still in its infancy. While there is intense activity in the area, most initial research is concentrated in adding power-awareness as a second-tier design goal which complements the more traditional real-time design goals. According to this approach, the system is first optimized subject to traditional real-time design constraints like timing and reliability. More often than not, an additional optimization step subject to power-aware design constraints is then piggybacked to this design. We believe that power-awareness should be one of the primary design goals for real-time systems, integrated in the design process at all levels, simultaneously coexisting with the traditional real-time design objectives. This requires a radical rethinking of the design methods as well as the definition of new metrics, a vision that is already becoming more ingrained in the research community.

II. Tutorial on System Level Power-Aware Design

A. Device-Level Power Dissipation Basics

CMOS devices are currently the building blocks for computing elements used in real-time systems. We distinguish between three types of power dissipated in CMOS circuits: *dynamic*, *static* and *short-circuit*. Short-circuit power is dissipated when both the NMOS and PMOS transistors are conducting simultaneously, and is due to the flow of current from power source to the ground. However, short-circuit power is comparatively insignificant and therefore is not within our scope. Dynamic power is consumed due to the switching of gates and is still responsible for a large percentage of the total power dissipated in current computing devices, although power dissipation related to static power is expected to increase in the future. Dynamic power reduction techniques form the bulk of the research reviewed in this survey. The average dynamic power is given by the following formula:

$$P_{DYNAMIC} = C_L \cdot N_{SW} \cdot V_{DD}^2 \cdot f \tag{1}$$

where C_L is the load capacitance, a substantial portion of which is wire-related, N_{SW} is the average number of circuit switches per clock cycle, V_{DD} is the supply voltage and f is the clock frequency. The capacitance C_L could be reduced by physical design techniques. Using place and route optimizations that strive to shorten or eliminate long wires with high capacitative load helps to decrease the impact of this term. Decreasing the second term, N_{SW}, the average number of circuit state switches, is achieved using techniques like minimizing the Hamming distance in operations/instructions following each other [13] or minimizing the number of operations [14]. This term is targeted especially by the hardware/software codesign community [15]. Since the supply voltage V_{DD} contributes a quadratic term to power dissipation, it is being aggressively targeted at the device-level. As technology advances towards deep submicron devices, the shrinking of the feature size enables a decrease of the supply voltage. According to ITRS projections [16], the 20 nm transistors that will go into production around 2007 are expected to run at 0.75 volts.

Unfortunately, decreasing the supply voltage increases the circuit delay, δ, given by:

$$\delta = \frac{C_L \cdot V_{DD}}{K \cdot (V_{DD} - v_T)^{\alpha}} \quad (2)$$

where K is a process dependent constant, v_T is the threshold voltage and α is another process dependent parameter which varies between 1 and 2. One can see that the closer V_{DD} gets to v_T, the higher is the delay. It is obvious that V_{DD} and v_T have to be scaled down simultaneously in order to maintain performance while saving power. Therefore, some device-level researchers have proposed and built circuits which have either variable or multiple v_T's and V_{DD}'s. Due to the increased delay, the circuits could not be driven at the same frequency f when the supply voltage V_{DD} is decreased. Consequently, the frequency must be reduced as well. Many power-aware real-time/embedded processors provide runtime means to decrease the supply voltage and the clock rate. This scheme, termed Dynamic Voltage Scaling (DVS), is employed extensively at the system-level. DVS works by slowing down the system when maximal performance is not needed. Most approaches differ in the way they decide *when* and *for how long* it would be profitable to apply voltage scaling.

The third component of the power dissipation, static power, is mainly due to leakage current between the power supply and ground. Leakage current has five components: reverse biased pn junction current, sub-threshold leakage, gate induced drain leakage, punch through and gate tunneling. Subthreshold leakage current dominates the leakage current and is in turn dominated by temperature and threshold voltage. Specifically, subthreshold leakage current scales exponentially with decreasing threshold voltage. Thus, lower threshold voltages lead to increased subthreshold leakage current and increased static power. As such, static power is independent of the circuit activity but dependent on the device area and temperature. Therefore, static power consumption occurs as long as power is supplied to the CMOS device. Although static power consumption is insignificant in current device sizes, it is expected to dramatically increase with shrinking feature sizes in the future. There is already considerable research activity to decrease static power consumption for general-purpose computing systems by turning off the voltage supply to portions of the chip that are not being used.

Shrinking feature sizes are responsible for increasing thermal-related problems as well. The on-chip temperature in current processors can vary by as much as several tens of degrees from one portion of the chip to the other with the maximum temperature reaching as high as 100^o C. The maximum chip temperature, T_{max}, is related to the chip area and maximum power dissipation, both of which tend to increase with scaling:

$$T_{max} = T_{amb} + R_n \frac{P_{max}}{A} \quad (3)$$

where T_{amb} is the ambient temperature, P_{max} is the maximum power dissipation and A is the chip area. R_n is the equivalent thermal resistance of the packaging and substrate (Si) layer. Reducing the term of interest at the system-level, maximum power, will also reduce the maximum temperature. Device-level related thermal issues are covered in [17]. For detailed information about device-level power dissipation basics, we refer the reader to [18], [19], [20], [21].

B. What is System-Level Power-Aware Design?

All the techniques for system-level dynamic power-aware design focus on one or more of the terms in Equation 1. This is the case even for C_L: the load capacitance may seem to be highly dependent on the physical design process and not affected by system-level design optimizations. However, consider an architectural design employing Globally Asynchronous Locally Synchronous (GALS) principles [22]. GALS is a clustered design with each cluster having its own clocking domain. Thus, compared to a completely synchronous architecture, the shorter wire lengths in GALS decrease the C_L. A power-aware compiler might employ instruction scheduling to decrease N_{SW}. An OS-level heuristic might then scale down the frequency f and voltage V_{dd} when peak performance is not required. A network layer scheme might put the network interface into standby mode when it is not likely to receive a message, thereby eliminating dynamic power dissipation for the duration. Therefore, power-awareness is embedded in every step of the system design hierarchy. The answer to the above question is now obvious:

System-level power-aware design includes power and energy management and modeling issues at the *microarchitectural, compiler, operating system (OS) and networking* layers. By contrast, device-level power-aware computer design covers power issues at the device and circuit level. Up until the mid 1990's, power-aware design was restricted to device-level low-power design. Architects, compiler and OS experts and network engineers designed for speed and performance; power and energy was considered to be the responsibility of circuit and layout engineers. However, in the last couple of years there has been explosive interest in establishing power and energy as important design goals at the higher layers of computer systems. The reasons behind this phenomenon are explained next.

C. Why is System-Level Power-Aware Design Important?

After all, a persuasive counter claim could be made as follows: CMOS circuits are the basic building blocks of the current microprocessors. Dynamic power in CMOS circuits is dissipated according to Equation 1. When circuit technology scales through shrinking the transistor feature size by a factor of x, the capacitance is reduced by x and the supply voltage by x^2. Therefore, power decreases by a factor of x^3, provided the frequency remains the same. Consider shrinking the transistor feature size from $.18\mu$ to $.13\mu$, thus scaling by a factor of 1.38. This means that power will be reduced by $1.38^3 \approx 2.6$ times. Since power seems to be significantly decreased by each new generation, there is no need to be concerned with power at the higher levels of system design.

Unfortunately, this is no longer the case. With each generational scaling of the feature size, more complex, aggressive designs are used. These designs employ higher

clock frequency, larger chip area and higher total number of transistors due to the use of more aggressive speculative execution. The result is a significant *increase* in power dissipation. Consider Table I: although the power supply voltage has decreased, total power dissipation has increased fivefold over the lifespan of the Alpha processor family. On the other hand, aggressive, complex designs increase the opportunities available for power management: there are more individual units which can be placed on standby when not needed by the application.

Processor	Power (Watts)	Freq. (MHz)	Die Size (mm^2)	V_{dd}
21064	30	200	234	3.3
21164	50	300	299	3.3
21264	90	575	313	2.2
21364	100	1000	340	1.5
21464	150	2000	396	1.2

TABLE I
CASE STUDY FOR SCALING: THE COMPAQ ALPHA (SOURCE:WILCOX AND MANNE [23])

Another worrying trend is the increase in power density. Consider Figure 1 for the Intel family of microprocessors. The power density is expressed in terms of watts/cm^2: the current generation is getting close to the power density of a nuclear reactor. Note that the power density of a hot plate has already been exceeded. This results in more expensive cooling mechanisms and reduced reliability. The increase in total power dissipation as well as power density means that traditional power management policies centered only at the device and VLSI levels are no longer sufficient. As a result, power has propagated as an important design constraint to the higher levels. The reader can find further information and surveys regarding system-level power-aware design in the following articles [25], [26], [27], [28], [29], [30], [31].

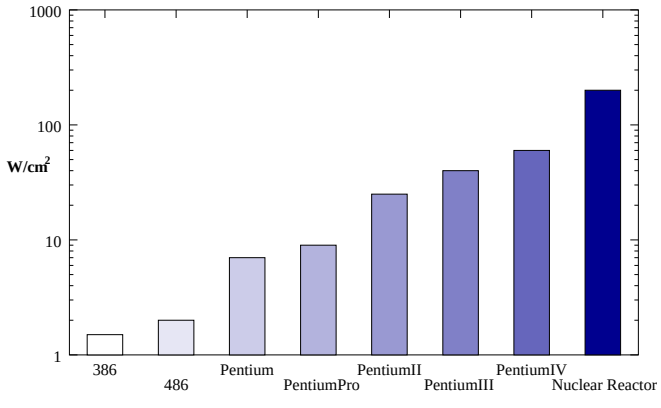


Fig. 1. Power Density of Intel Chips (Source: Fred Pollack [24])

III. LAYER BY LAYER SURVEY ON SYSTEM LEVEL POWER-AWARE REAL-TIME RESEARCH

Processors used in real-time systems are usually more conservatively designed. The typical real-time processor, if pipelined at all, can issue a single instruction per cycle and instructions execute in-order. Although not very common lately, even caches used to be disabled. The reason for this restraint is threefold; first, the use of a time-tested and provably correct architecture provides a reliable foundation block for the system. Second, the simple architecture makes the incorporation of fault-tolerance such as hardware redundancy easier. Third, this simplicity offers ease of modeling for WCET analysis. It is worth noting that even simple pipelining is quite difficult to model [32]. The same reasoning provides the rationale behind disabling caches.

However, it is our belief that more complex architectural features such as multiple out-of-order instruction issue or multithreading will trickle down to processors used in real-time systems and offer more research opportunities. Currently, real-time systems research at the microarchitectural level for power savings is concentrated on the following issues: instruction set architecture selection, instruction caches and the system bus, cache region reservation, voltage and frequency scaling, battery-consciousness, Field Programmable Gate Arrays (FPGA), and task movement. We consider each of those topics next.

Instruction Set Architecture (ISA) Level: This is an active research area in the context of general-purpose architectures; various researchers have commented on the need to take power and energy into account in ISA design. However, not much effort has been devoted to power-aware ISA design in real-time systems. We now briefly discuss one of the few representative works in power-aware ISA issues in real-time systems by Cheng et al. [33]. The authors employ a fine-grained off-line scheduling approach which saves power by combining multiple instructions into one complex but lower power instruction or by using low-power versions of instructions while considering task deadlines. The proposed scheme assumes that the ISA is sufficiently flexible, however, in practice there is not much scope for the existence of complex instructions which are functionally equivalent to a group of simpler instructions in the ISA design.

I-cache and Buses: The control path, which governs the fetch, issue and retiring of instructions, is quite simple in typical embedded processors and occupies a relatively small portion of the chip area. The caches take up most of the chip area [34] and are responsible for a considerable percentage of the energy dissipation even though memory is more energy efficient than control logic. We concentrate here on the instruction-cache (I-cache) and consider two representative power reduction approaches. Benini et al. [35] compress the instructions in memory. A compressor/decompressor is inserted between the memory and the CPU and therefore no architectural changes to the ISA are necessary. The observation is that a subset of the ISA is used for programs so the most frequently used instructions are compressed. This saves instruction fetch energy by using fewer bits on a fetch. An alternative strategy by Lee at al. [36] also concentrates on saving instruction energy. The authors employ a loop cache and keep the tight loop in a small loop cache instead of accessing a larger block. The power savings come from two factors: first, less power is consumed per access by accessing

a smaller loop cache instead of a larger monolithic cache; second, unlike regular caches, the loop cache has no address tag store and no valid bit, thus saving power. The enabling factor is a special class of branch instructions, called short backward branch instructions, that were developed specifically for loops. This work also demonstrates the usefulness of augmenting an ISA in a power-aware fashion for real-time systems. Besides being energy-efficient, the loop-cache also enhances predictability in real-time applications by keeping frequently executed tight loops in the cache.

Power-aware bus encoding techniques decrease the off-chip power dissipation of address buses. A representative example is the bus-invert scheme proposed by Stan and Burleson [37]. In this scheme, the number of bit transitions that might occur with respect to the previous data are computed before the data is put on the bus. If the transition count is more than half the bus width, the data is inverted and put on the bus, thereby saving power. The inversion of the bus is signaled through an extra bit line. More recently, Mamidipaka et al. [38] observed that bus transitions are an important source of power dissipation in embedded systems. They proposed a low-power address encoding scheme using self-organizing lists which exploit the spatial and temporal locality of the addresses. This approach saves power on the data busses (up to 54%) as well as address busses (up to 59%).

Cache Region Reservation: Since many real-time applications have tight memory requirements as well as power and timing constraints, a natural optimization approach is to reserve regions in the cache. For real-time systems, there is a strong research tradition for cache/memory region reservation techniques. However, most of the previous approaches considered locking or reserving memory to increase execution time predictability and not to reduce power. One of the first efforts to consider power is by Li et al. [39]. Their aim is to find a data or instruction cache configuration so that the real-time constraints can be met with the lowest power. They assume a single CPU with multiple real-time tasks running at different rates. The scheduling algorithm is the Earliest Deadline First (EDF), a widely used preemptive algorithm in real-time systems. In EDF, the task that has the earliest deadline in the time domain has the highest priority. It has also been proved that if the total system load is not greater than 100% and task deadlines equal their periods, then the task set is schedulable on that processor [40]. If the utilization is more than 100%, then caches are added to decrease the WCET of certain tasks until the utilization drops below 100%. Li et al.'s partitioning heuristic sets aside cache partitions for each task. If a task's instructions or data are reserved a partition, then the instructions or data will always stay in the cache. This ensures that the program always has instruction or data hits. The partition allocation heuristic uses a linear programming approach for region allocation. The optimization procedure considers the total cache size and task schedulability as constraints and solves for minimum system power under these constraints.

Voltage and Frequency Scaling: Real-time systems are typically over-designed, provisioning resources for the *worst-case* execution time (WCET). Since tasks rarely execute up to their WCET, there is significant scope for power and energy savings using dynamic voltage and frequency scaling (DVS). The processor must be augmented with hardware blocks that allow changing the supply voltage dynamically, see Figure 2. Reduction of V_{dd} and/or frequency saves substantial power, however it also increases the circuit delay, causing a slowdown in the execution of programs. Therefore, DVS heuristics usually trade off power savings against delay. One of the earliest papers in this area is by Yao et al. [41]. The authors consider a set of tasks with an identical period. They define the average-rate requirement of a task to be the ratio of its required number of cycles divided by its time frame (deadline−arrival time). At time *t*, the energy-saving heuristic sets the speed of the processor to the sum of average-rate requirements of tasks which are in the same time frame. Kirovski and Potkonjak [42] extend the analysis to tasks with arbitrary periods and explore processor allocation as well as task assignment for voltage scaling. All tasks are assumed to be independent. The allocation step finds the most energy-efficient number of processors to use given the task set. Steepest descent heuristics are used for allocation and task assignment.

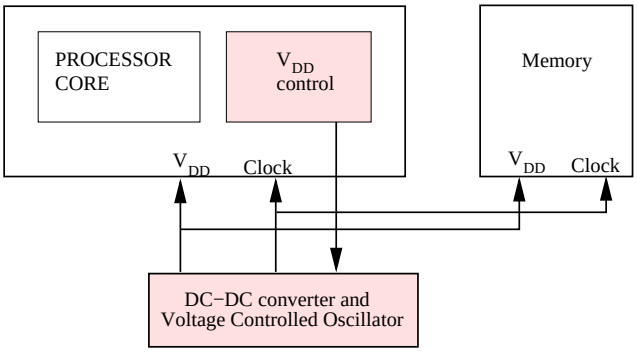


Fig. 2. Augmented processor blocks to implement dynamic voltage/frequency scaling

Lee and Sakurai [43] also use DVS, but the supply voltage changes are allowed only at specific points during the execution of a task. They partition each task into time slots. This is useful in processors employing DVS strategies at specific intervals. Kang et al. [44] also vary the supply voltage but they consider task dependencies. They use task chains for modeling and target distributed real-time systems. Krishna and Lee [45] adopt a two phase heuristic that has offline and a steepest descent-based online components. Luo and Jha [46] consider aperiodic tasks as well, which are assumed to be soft real-time. The scheduling scheme is static for the hard real time periodic tasks and dynamic for the soft aperiodic tasks. Kirovski and Miodrag [47] prove that voltage scalable task allocation and scheduling optimization problems are NP-complete.

Scaling the frequency and voltage takes a non negligible amount of time, which can be in the order of tens of microseconds. This overhead is considered by AbouGazeleh et al. [48]. Hong et al. [49] use synthesis techniques with dynamic voltage scaling. The heuristic computes power efficient I- and D-cache sizes. Other papers that are based on DVS schemes include utilizing integer linear programming to minimize power [50],

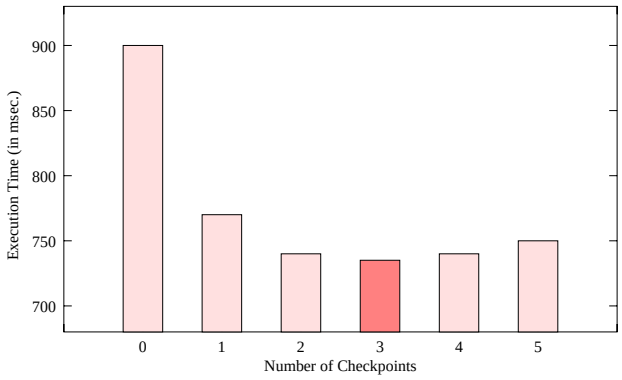
considering fixed-priority scheduling [51], extensions for non-preemptive [52] and preemptive aperiodic tasks [53].

Battery Conscious Real-Time: Many real-time platforms, such as space-borne systems use batteries that constrain available energy. The most important issues to be considered here are the total battery capacity, expressed in Ampere-hours or Watt-hours and the battery discharge profile. The latter is important in devising battery-aware schemes that are guided by the discharge profile. In one such recent work, Luo and Jha [54] consider distributed real-time systems and develop a battery model, which is used in two scheduling schemes: first they optimize the battery discharge power profile, and then they use voltage scaling for distributed real-time systems. The overall objective is to extend the battery lifespan while meeting task deadlines and precedence requirements. The authors claim that mitigating battery capacity loss requires reducing the discharge current level and shaping its distribution. The first scheme employs schedule transformations. Starting from a valid schedule, the transformations are based on minimization of the peak power consumption and reduction of the variance of the discharge current profile. The second scheme is based on slack time reallocation which reduces the average discharge current level. In the case of systems with more than one means of power delivery, a different metric is needed. In their approach, Liu et al. [55] study the Mars rover application. They develop a maximum power-budget concept and propose heuristics similar to [54] for a dual power-source system. A power-usage metric (free vs. expensive) is developed from a depletion point of view: free solar power cell and an expensive battery based-power source.

FPGA for Real-Time: FPGA-based systems are considered to be less energy efficient compared to other architectural alternatives but they offer significant potential for power-reduction techniques since many blocks of the FPGA could be inactive and still consume power. Park and Burleson [56] examine reconfigurable architectures and consider a real-time video application. One suggestion for higher energy-efficiency is to utilize free FPGA resources as local memory to avoid off-chip communication. Another suggestion is to avoid unnecessary computation by adjusting the search area according to the changing characteristics of the video signal.

Task Movement: Task movement is important in real-time systems for fault-tolerance or load balancing purposes. However, power efficient task movement heuristics have not been extensively investigated. One exception is the work of Liu et al. [57]; this is based on the observation that a set of processors can operate at a lower power level than a single one with the same performance if there is enough parallelism.

Fault-Tolerance: Including power-awareness in fault-tolerant systems may lead to slightly divergent design points from that of non-power-aware schemes. We illustrate the above assertion with a simple example: consider the problem of optimal checkpoint placement to recover from transient faults. We want to find the energy-optimal strategy. During checkpointing, the power dissipated per time unit is higher than during regular program execution. Since the stable storage has to be accessed very frequently, energy is consumed at a higher rate. Assume for example, that running the application without any checkpoints consumes 120 millijoules of energy and takes 600 milliseconds to complete. Also assume that there is a single transient fault during execution. The energy consumed during the checkpointing process is 10 millijoules and each checkpoint takes 20 milliseconds. The energy consumed to recover from the transient fault by rolling back to the last checkpoint and resuming execution is half of the inter-checkpoint energy on average. A traditional non-power-aware placement strategy, shown in Figure 3a, would minimize the overall execution time. According to this criterion, the optimal number of checkpoints is 3. However, if the criterion is minimization of energy, shown in Figure 3b, the optimal number of checkpoints is 1 or 2. Thus, time and energy optimal solutions have different design points. A power-aware design which takes timing constraints into consideration would choose to employ 2 checkpoints, which is energy-optimal with minimal execution time degradation.



(a) Non-power-aware placement

(b) Power-aware placement

Fig. 3. Time- and energy-optimal checkpoint placements.

We conclude the survey of the microarchitectural layer with a short synopsis of power-aware soft real-time research. Hughes et al. [58] evaluate the relative merits of architectural and dynamic voltage scaling techniques for power reduction in multimedia applications. A variety of architectural configurations are studied with distinct instruction window, issue

width and functional unit (FU) sizes. Generally, the architecture scales from a very aggressive 8-wide issue, 128-entry instruction window, 8 FU configuration to a less aggressive processor with 2-way issue, 32-entry instruction window and 4 FUs. The developed scheme operates at the frame granularity, and requires a profiling phase. The profiling phase predicts the energy per instruction for all possible architectural and DVS-based configurations using the profiles of a single frame. Then, before the execution of each frame, the adaptation control algorithm predicts the number of instructions that will be executed for the frame. Using the energy and instruction estimates, the algorithm chooses the most energy efficient architecture and DVS combination for each frame. The results indicate that the mean fraction of missed deadlines over all the applications is about 2.2%. One somewhat surprising conclusion is that, with DVS, more aggresive architectures are more energy efficient. Using an MPEG application, Martin and Siewiorek [59] determine that a processor speed-setting policy should consider battery capacity and memory bandwidth, more specifically, that the memory subsystem should be a major design consideration so that it does not limit performance over the range of CPU frequencies. The experiments were carried out through actual measurements with the important finding that both factors must be simultaneously considered when implementing voltage scaling. Along the same lines, these authors explore in [60] the impact of the non-ideal battery characteristics on power policies in a wearable computing setting.

A. Compiler Level

There has been very limited work in the area of power-aware compilers for hard real-time systems. This is a difficult task since the compiler optimizations would need to consider power and energy minimizations while making sure that those transformations do not violate timing constraints. Approaches at this level are mainly confined to soft real-time systems with most of the research focusing on embedded systems. The objective for these applications is to optimize memory usage since embedded systems are severely memory constrained. Here, saving power is a side effect of memory size and usage optimizations. We concentrate here on two subareas:

Voltage and Frequency Scaling: Azavedo et al. [61] develop a compiler-driven scheme for DVS that uses "checkpoints". The term checkpoints refers to statically determined program epoch points where processor frequency and voltage can be changed. The checkpointing information also specifies per-epoch timing constraints, so the scheme is fine grained at the intra-task level. User defined checkpoints are inserted into the source code and compiled into special instructions. The heuristic requires a profiling stage which reports per-checkpoint minimum and maximum power/energy/performance information.

Memory Organization: Levy et al. [62] note that the memory subsystem is responsible for up to 50% of total power dissipation in embedded systems. The main focus of the paper is the overview of compiler-driven power-aware research opportunities/perspectives for the memory. They identify specific issues for power-aware embedded memory systems such as block granularity, cache replacement algorithms, data clustering, and heap data allocation.

There is a rich tradition of compiler-driven power-aware memory exploration for soft real-time systems. An interesting analysis in [63] includes a negative result regarding the energy-efficiency of standard compiler optimizations: the application is an MPEG decoder on a battery-powered embedded system (Smart-Badge from HP labs, a wearable system) and their results indicate that standard compiler optimizations result in less than 1% energy savings, but source code level optimizations are capable of up to 90% energy savings.

Grun et al. [64] look at categorizing accesses for memory customization for embedded systems and their approach requires a profiling run. The authors utilize the results of the profiling to customize the memory architecture for different access and locality patterns, in effect partitioning the monolithic cache. The variables are clustered according to different types of locality. The power is reduced by using temporal caches (caches with small line size or overall size) for variables with high temporal locality and spatial caches (with large line sizes to exploit the spatial locality) for variables with high spatial locality. They report a 30% memory power reduction without reducing performance.

Shiue and Chakrabarti [65] develop off-chip memory assignment schemes to avoid cache conflicts and save energy. They define two arrays in a loop body to be compatible if they are independent of the loop index. If all accesses in a loop are compatible, then the conflict misses can be avoided by a suitable data layout organization in memory that would place compatible items in the same cache line. They also analyze the impact of tiling on power efficiency. Specifically, they examine the impact of tiling size. Tiling divides the iteration space into tiles and modifies the loop nest to iterate over them, reducing the miss rate and power. They consider a simple example as shown in Figure 4. Here, array b has stride 1 while array a has stride n. Interchanging will not help in this case since this would change the stride of array b to n. Using tiling, the miss rate is reduced from 0.44 to 0.22 for a tiling size of 2.

Kulkarni et al. [66] study techniques for power efficiency by program transformations while preserving real-time performance constraints. The applications studied are MPEG-2 decoder, QSDPCM video codec and a Voicecoder. In another paper, Kulkarni et al. [67] develop code transformations for embedded multimedia and DSP applications targeting power savings in the cache. They consider locality of data, size of data structures, access structures of large array variables, regularity of loop nests and the size and type of cache in developing the transformations. Soudris et al. [68] investigate power-aware data reuse techniques for real-time multimedia applications and determine that certain data sets are heavily reused in a short period of time. The reused data is then assigned to smaller on-chip memories which require less power per access.

Unsal et al.[69], [70] employ data partitioning based on type information. The authors find that scalars in multimedia applications have a small memory footprint but a high access frequency. Scalar accesses are mapped to a small scratchpad

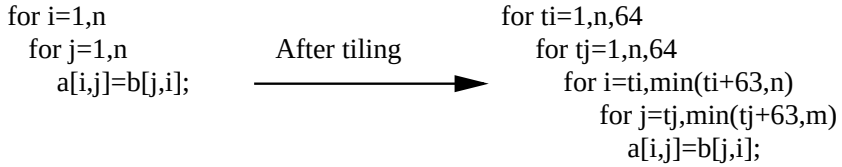


Fig. 4. A simple example for tiling (Source: Shiue and Chakrabarti [65])

SRAM area [69] or alternatively a minicache [70]. This small SRAM or minicache are more energy efficient than the large L1 data cache. They also introduce a compiler-controlled tagless caching framework. This compiler-directed cache is a flexible data cache that replaces the tag-memory and cache controller hardware with a compiler-managed tag-like data structure that is much smaller than a traditional data cache tag and saves energy in the data cache. Consider for example, the system shown in Table II. Here the scalar accesses are directed to a small minibuffer of size 1K. Assuming that the unpartitioned monolithic D-cache has a miss rate of 2%, it has been shown that the partitioned cache is 25% more power efficient than the unpartitioned one. Moreover, the smaller cycle time that is made possible by using a smaller structure helps the real-time performance as well.

Parameter	Attribute
64K L1 D-Cache power	3.0 Watts
D-Cache access time	2 cycles
D-cache miss rate	1.0%
1K Minicache power	0.5 Watt
Minicache access time	1 cycle
Minicache miss rate	4.0%
Minicache access percentage	35%
Main memory power	5.0 Watts

TABLE II
DATA CACHE CHARACTERISTICS FOR THE EXAMPLE

B. Operating System (OS) Level

Real-Time Operating Systems (RTOS) have been traditionally designed for modularity, many employing small *microkernels*; the real-time system engineer can then add modules as required (the networking module, serial communication module, etc.). This characteristic itself is naturally amenable to power-aware computing: unnecessary modules could be discarded, decreasing the power consumption. Typical issues to be considered at this level include: DVS, I/O devices, actual measurement of which blocks are responsible for most of the power dissipated due to OS activity, and soft real-time systems. We consider each in the following subsections.

Voltage and Frequency Scaling: Pillai and Shin [71] have introduced OS-level heuristics and have implemented them. The heuristics are based on the observation that most real-time tasks finish before their WCET, creating a slack which could be exploited for power-savings. The heuristics then scale the voltage and frequency while maintaining real-time deadline guarantees. The authors consider periodic, non-sporadic and independent real-time tasks. The heuristics are tightly integrated with the operating system and are implemented as extension modules to the Linux 2.2.16 kernel. A probe-based power measurement framework records the energy saved.

I/O Devices: Swaminathan et al.[72] study OS-directed power-aware I/O device scheduling for hard real-time systems. They propose an online algorithm which takes a predetermined task schedule and device-usage list as inputs and produces a sequence of sleep/working states for each device. The tasks have deadlines but are not periodic and are assumed to be independent. They show that energy-optimal device scheduling for real-time task sets is not possible without future knowledge of device requests.

Power and Energy Analysis of RTOS: The first effort in this area is by Dick et al. [73]. The authors analyze the power profile of a commercial RTOS, the μC-OS, by running two applications on a Fujitsu SPARClite-processor-based embedded system, and by evaluating the power consumed by the operating system calls. They show that the RTOS can consume a significant part of the system power and affect the power consumed by the other software layers. Baynes et al. [74] also analyze the μC-OS as well as two other OS's: Ecnidna and NOS. The NOS is a simple "bare-bones" scheduler and serves as a baseline case for comparison. The results show that the RTOS overheads are a factor of two to four times higher compared to NOS. Another key finding is that poorly designed idle loops could cause the system to double its energy consumption. Acquaviva [75] et al., on the other hand, aim to establish the power profile of an RTOS independently of the running applications. They experiment with various OS components and report that increasing the context switch frequency from 0Hz to 10 Khz does not affect the energy consumption, concluding that the context switch mechanism of an RTOS is energy-efficient. However, including the effects of flushing the cache during a context-switch increased the energy overhead. They also experiment with I/O drivers, specifically with the CPU sending data bursts that are large compared to the output buffer. The authors find that a considerable amount of energy is spent by the OS when the buffer is full, or when it polls a synchronization variable, similarly to [74]. They consequently suggest that an energy-aware RTOS should send data in smaller chunks, if possible.

Distributed Real-Time Systems: Unsal et al. [76] examine high-level allocation and scheduling heuristics for distributed hard real-time systems. Both tightly- and loosely-coupled systems are considered. For loosely coupled systems, the authors propose a power-aware shared data structure allocation algorithm which clusters tasks using the particular shared data structure to the same or a topologically close processor, thereby saving power. Since the assignment problem is NP-complete, they develop a simulated annealing approach to

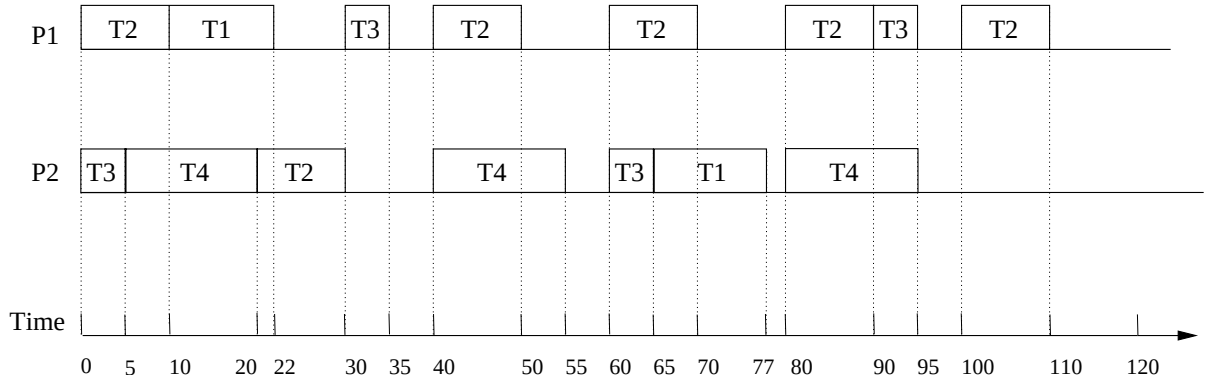


Fig. 5. Execution timeline of the tasks for the example (PI denotes Processor I)

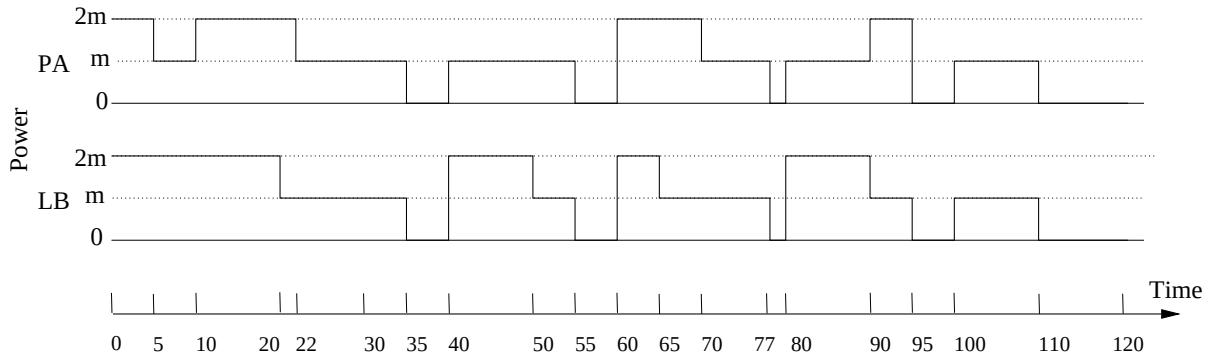


Fig. 6. The tasks that are active at the same time and the associated memory bank power consumption for the different schemes (LB: Load-Balanced PA: Power-Aware).

obtain solutions that are close to optimal. For tightly-coupled systems, the authors examine the problem of allocating tasks to memory banks. Energy is saved by turning off memory banks that are not needed. This is maximized through a heuristic which assigns tasks whose execution times overlap to a large degree, to different memory banks.

Task id	Execution Time	Period	Utilization(%)
T1	12	60	20.0
T2	10	20	50.0
T3	5	30	16.6
T4	15	40	37.5

TABLE III
THE TASK SET FOR THE EXAMPLE

The example below illustrates the contrast between classical resource-driven task allocation policies and a power-aware one. The system under consideration is a tightly-coupled distributed real-time system with two processors and memory banks. There is a single run-queue. In this setup, the tasks are run on the first available processor and are assigned to memory banks, i.e., a single memory bank is selected to hold the data and text section of a task. If only X bank(s) are used by the application tasks at a given time, the remaining $2 - X$ bank(s) can then be put into sleep mode to conserve energy. We are only considering dynamic power, so there is no power dissipation when a memory bank is in sleep mode. The delay caused by simultaneous accesses to a memory bank by two tasks is negligible. There are four periodic tasks and their execution times, deadlines, and utilizations are shown in Table III. Suppose that the memory utilization of each task is linearly dependent on its execution utilization. The multiprocessor scheduling heuristic is earliest-deadline-first adapted to the single run queue. In this variant, the chosen task is allocated to the first available processor. The resulting execution timeline of the tasks on the processors is shown in Figure 5. Note that the chart remains the same independent of the allocation of tasks to memory banks so the resulting processor related power consumption is the same and is not the main focus here. However, the power consumption of the memory banks is dependent on the allocation of the tasks to memory banks. Two tasks are allocated to each memory bank. The memory bank related power consumption per time unit is $2m$, m, 0 if two banks are active at a time instant or if one bank or none are active, respectively. A classical distributed allocation scheme would try to balance the load of the memory bank utilization. This load balancing (LB) scheme may assign tasks T1 and T2 to bank 1 (bank utilization 70%), T3 and T4 to bank 2 (bank utilization 54.1%). A power-aware (PA) scheme, on the other hand, would try to assign harmonically-related tasks to the same bank. The intuition being that by

allocating tasks whose periods are multiples of each other, the tasks would be simultaneously active more frequently over the execution timeline of the tasks and therefore save power. Following this scheme we allocate T1 and T3 to bank 1 (with periods 60 and 30, bank utilization 36.6%), T2 and T4 to bank 2 (with periods 10 and 40, bank utilization 87.5%). Note that PA is more load imbalanced than LB. The resulting bank power consumptions are shown in Figure 6. The total memory bank power consumption of the LB scheme is 137m, while that of the PA scheme is 124m; a power savings of 15%.

Soft Real-Time Systems The OS-level research in power-aware soft real-time systems is gaining in importance. This is driven by the handheld or pocket computers which are becoming more popular and feature such soft real-time applications as handwriting recognition, audio and video playback and Global Positioning System (GPS). Farkas et al. [77] examine the power profile of a pocket computer and show that it exhibits a wider range of dynamic power than a notebook computer. The authors then examine the energy implications of various Java design options. Since the Java Virtual Machine (JVM) exists as a middleware layer between the application and the OS layer, this analysis has OS implications. They report that using a single JVM (that the users would share) is 25% more energy efficient than using one JVM per application. Another key finding is that techniques such as just-in-time compilation, which aim to minimize overall execution time, are energy-efficient. Yuan and Nahrstedt [78] also focus on a middleware approach but they consider applications running on a laptop instead of a pocket computer. The authors develop an Advanced Configuration and Power Interface (ACPI) compliant processor/power resource management scheme. They implement the scheme on a Windows NT platform running on a laptop. The applications are a math program with periodic constant processing time, and an MPEG decoder with periodic variable processing time. Based on a simple energy model, they claim 39.5% savings.

Batteries: Ma and Shin [79] examine energy-aware quality-of-service tradeoffs. To this end, the authors develop an energy aware scheduling algorithm which favors low-energy and critical tasks. Depending on user needs, the algorithm is tunable toward extended battery life at the expense of performance. Their simulation results show that battery life can be extended by up to about 100% with performance degradation of about 40%. Benini et al. [80] demonstrate open and closed-loop dynamic power management strategies on a digital audio recorder application. They stress the fact that policies that seek to minimize average power reduction need not neccessarily result in extended battery lifetime. Their objective is maximizing the time of operation of battery-operated portable equipment through utilizing the information about the battery's charge state. Flinn and Satyanarayanan [81] also target battery time and propose a synergestic scheme based on a tight cooperation between the applications and OS. The authors study video player, speech recognizer, map viewer and web browser applications. The results suggest that battery life could be extended by up to 30%.

Web Servers: The increased popularity of internet applications places an immense burden on the power costs of web hosting centers. Bohrer et al. [82] examine energy efficiency in web servers. The authors establish the motivation by observing that web server capacity is planned to provide acceptable service even in hours of peak demand. In other words, the design is for the worst-case similarly to hard real-time system design philosophy. Therefore, the web server might not be operating at its maximum capacity for long periods of time. Using actual data from the web logs of the 1998 winter olympics in Nagano, the authors note that while the peak workload was 1840 hits/second, the average workload was only 459 hits/second. This data indicates that on the average the site was operating at about 25% of the peak capacity. By predicting periods of low activity and employing the use of voltage and frequency scaling, they report that savings in the range of 23% to 36% are possible while preserving server responsiveness.

Jitter: Sometimes system behavior that is deemed to be harmful to performance can provide new approaches for a power-aware objective. Consider jitter, which is an undesirable property for streaming applications. While clearly irritating for the end-user, a jitter-prone soft real-time system can be leveraged for power savings. Jitter usually manifests itself in the form of early or late frame arrival. If a frame arrives early, then DVS schemes could be used to reclaim the slack for power savings. On the other hand, if a frame arrives late, then some relatively unimportant frames can be dropped thereby saving power in this case as well.

C. Network Level

Traditional real-time research on this level concentrated on designing communication protocols that provide bounded response times and therefore increase determinism. Including power-efficiency as a complementary goal presents an additional challenge. Qu et al. [83] consider energy minimization of deadline constrained communication systems. The communication system is modeled as a sequence of store-and-forward pipeline stages. The authors develop an algorithm to compute k, where k is the power-optimal number of fragments for a packet. The approach is then applied to the 4-stage Myrinet pipeline resulting in 27% to 93% power reduction depending on the pipeline stage. Gruenwald and Banik [84] study real-time databases and propose a transaction management framework that reduces deadline missing transactions while balancing the energy consumption by the mobile hosts in the system. Unsal et al. [85] consider energy-aware data replication on distributed real-time systems. The authors also study the energy impact of network topology and broadcasting in real-time systems and develop a power-aware real-time multicasting scheme which relies on a Steiner tree heuristic. Lee et al. [86] conduct a formal modeling for power-aware real-time systems based on process algebra. The proposed framework allows the modeling of probabilistic resource failures as well. The approach is applied to a power-aware ad-hoc network protocol.

The commercial/research opportunities for power-aware soft real-time systems have just started being tapped into. The introduction of the wireless WiFi (IEEE 802.11) standard has

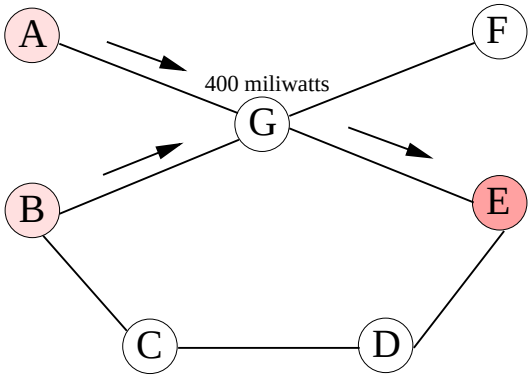
led to increased activity. The next two papers deal with power-aware issues for this standard. In the first paper, Qiao et al. [87] study combining transmit power control and physical layer rate adaptation for energy-efficient operation of the IEEE 802.11a wireless LAN. Lahiri et al. [88] discuss issues in the development of a battery-efficient 802.11 Medium Access Control (MAC) processor. As such it combines issues at the network and architectural layers. In the design process, the authors start with a 'raw' architecture, apply typical MAC frames to this architecture for current profiling, analyze the profile to identify regions where the battery is inefficiently discharged, identify which states of the processor are responsible for the bottleneck regions and modify the MAC processor bus protocol to decrease the bottleneck region. The authors reapply this Hardware/Software codesign refining process until the desired level of battery efficiency is reached. They examine energy savings for both streaming applications with QoS requirements as well as non-real-time applications.

An earlier work by Havinga and Smit [89] also considers the MAC layer. The authors concentrate on handheld computers and study power-aware network protocols. The MAC protocol maximizes the standby mode of network interfaces to reduce energy. In another paper the same authors [90] explore power-aware buffering strategies and an ATM-like switching fabric architecture for multimedia applications. They also discuss architectural issues such as clock gating toward power efficient operation of the switching fabric. Gomez et al. [91] concentrate on routing optimizations in wireless networks to save power. In their adaptive packet forwarding scheme, the intermediate nodes act as redirecting agents and perform route optimization which can lead to discovery of new routes that require less transmission power. The energy-efficiency of existing popular streaming media products have been analyzed in [92]. There, the author shows that Microsoft Media Player exploits network level fragmentation for high bandwidth streams which leads to wasted energy in a lossy network. However, since the packet transmission rate in Media Player is regular, this enables simple history-based client end policies to use lower power states with quite small data losses. The analysis of Real and Quicktime products reveals that transmitting larger data packets at regular intervals could increase their energy efficiency.

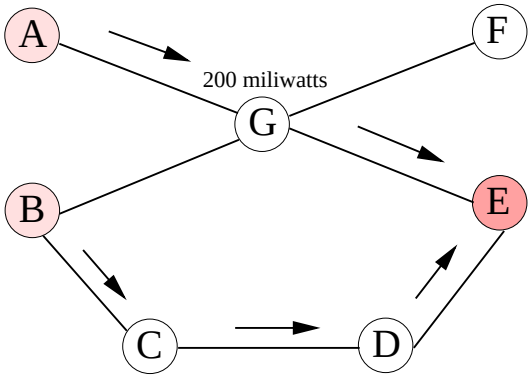
An emerging research area, especially for military applications, is wireless sensor networks. Sensor networks usually have low duty cycles, low-bandwidth communication, non-replaceable batteries and short transmission distances and they also have dynamic configurability and fault-tolerance requirements. Taken together, those characteristics call for a unique power-aware research agenda. Some of these issues are addressed in an overview paper on low-power wireless networks [93]. Presenting several key technologies required for low-power sensor network, the paper is a good introduction to the topic. Network concerns such as communication (at the physical, MAC, and network sublayers), routing and data forwarding are main research foci. We present here a short synopsis of power-aware network sensor work: Rabaey et al. [94] develop a model for packet forwarding which also includes a term for path-loss term. An interesting, somewhat counter-intuitive, observation that they make is that it might be more energy efficient to send a bit using several short hops instead than using one longer hop. Other researchers have looked into clustering issues: due to the short transmission distances, it might be necessary to form several clusters of sensor nodes. Moreover, it might be necessary to dynamically change the sensor network configuration from time to time. Therefore, a dynamic clustering approach would be more suitable for sensor networks. Hienzelman et al. [95] propose Low-Energy Adaptive Clustering Hierarchy (LEACH), a dynamic clustering approach. LEACH uses a randomized rotation of local cluster base stations (responsible for inter-cluster communication) to evenly distribute the energy load among the nodes. The routing protocol uses data fusion to minimize the information sent by the sensor nodes to the base station, reducing energy consumption. Their simulation reports reduction by a factor of 8 in energy dissipation compared to conventional routing protocols. The MAC layer in sensor networks has different design goals than traditional wireless MAC's such as the IEEE 802.11: energy conservation and self-configuration are more important than per-node fairness and latency. Ye et al. [96] consider those design goals in developing an energy-efficient MAC protocol, termed S-MAC. S-MAC reduces energy consumption by putting nodes to sleep while they are listening to an idle channel. Neighboring nodes form clusters to establish sleep schedules. S-MAC also sets the radio to sleep when other nodes are transmitting. Results indicate that S-MAC consumes 2-6 times less energy than an IEEE 802.11-like MAC. Tsiatsis et al. [97] study energy-efficient packet forwarding in wireless sensor networks. Each sensor node is composed of two blocks: the sensor node CPU and the radio board. The authors propose to migrate part of the network layer responsibility from the sensor node CPU to the radio board. Power is saved since packets are now processed by the radio board, instead of moving up to the sensor node CPU.

Maximum power reduction is an important design goal, especially for thermally-constrained systems. As an example we consider the goal of maximum power reduction for real-time message delivery in a router setting. In power-unaware routing, shortest path schemes usually provide the best performance, minimizing the average end-to-end message delay. However, these schemes create hotspots, which increase congestion and the maximum power consumed. Alternatively, by identifying and avoiding hotspots, maximum power can be decreased. Note that this might mean that the total energy consumption will increase. For motivational purposes, consider a 7 processor system as shown in Figure 7. Nodes A and B simultaneously send a message to node E. Assume that the deadline for the messages to be received at node E is 20 μseconds. Assume further that it takes 5 μseconds for a message to traverse a hop and that the router power dissipation at a node is 200 milliwatts. As shown in Figure 7a, the maximum power dissipation in a power-unaware scheme based on shortest-path routing is 400 milliwatts (at node G) and the messages are received at node E in 10 μseconds. A power-aware scheme utilizes a different path for the message from node B to E (see Figure 7b), with E receiving the

last message in 15 μseconds; still less than the deadline. Moreover, the maximum power dissipation is now reduced to 200 milliwatts, an improvement of 100%. A generalization of this example would be a topology-aware, slightly randomized shortest-path routing scheme. This scheme would take more aggresive randomized routing decisions around the hotspots, subject to message deadlines being met, trading off maximum power reduction for message delivery performance.



(a) Non-power-aware routing



(b) Power-aware routing

Fig. 7. Shortest-path versus maximum power reducing power-aware routing.

IV. Epilogue

We have presented an overview of system-level power-aware design methods in real-time systems. Where appropriate, we included examples and challenges. Our goal is to ferment further research in this evolving field. In the same vein, we firmly believe that new paradigms and modification of current design techniques need to be developed from the earliest stages of real-time system design, with power-awareness embedded as an indispensable design objective. We conclude with several examples that support this argument.

Example 4.1: On a meta level, reliable real-time system design requires a well-defined interface between the user, control engineer and the computer architect. This interface is specified using *performability*. An amalgam of performance and reliability, performability defines several accomplishment levels, which pertain to the different performance levels as perceived by the user. Performability of a real-time system is thus the probability that the design will allow each accomplishment level to be met. At the high-level design, power-awareness (in the form of energy budget, maximum power requirements or any other power-aware measure) could be embedded as one or more accomplishment levels. This addition will then provide valuable probabilistic information as to the real-time system's ability to meet each power-aware accomplishment level.

Example 4.2: Here we demonstrate the need for new research in execution time estimation to support power-awareness. Traditionally, WCET analysis was sufficient to satisfy the performance requirements for real-time systems, there was no need to provide the complete execution time histogram. Now consider two applications (see Figure 8), and a DVS scenario; both applications have the same WCET. However, DVS schemes take advantage of tasks that finish *before* their WCET. Therefore, the application in Figure 8b has more "capacity" for power-savings through DVS than the one in Figure 8a, since the average execution time of the task is comparatively lower. An execution time analysis based on the extraction of WCET would not be able to capture this potential, compared to one which analyzes the *distribution* of execution times.

Example 4.3: Load balancing is an important research area in the domain of distributed real-time systems. More suitable to distributed shared-memory-based systems, traditional load balancing attempts to minimize the task response time by dynamically reallocating tasks to processors subject to deadline feasibility constraints. However, this approach might not decrease the total energy consumption. Consider two tasks, with similar WCET. The rate of energy consumption of one task could be significantly higher than the other due to a higher circuit activity factor such as more frequent memory accesses. Therefore, one needs to consider the activity factor of tasks. An "energy balancing" scheme would then utilize this information to minimize the total energy consumption, subject to deadline constraints.

System-level power-aware research for real-time systems is a new and vibrant research area. Many current approaches make use of dynamic voltage scaling (DVS) heuristics and slack reclamation for power and energy savings. We believe that the scaling down of supply voltages and the fundamental limits on threshold voltage reduction will limit the gain from DVS in the future. More diversity in novel power-aware real-time design techniques, especially in the domain of fault-tolerance will be needed. Static power reduction by predictively turning off inactive blocks and functional units will also become more important, with the additional requirement of preserving timing constraints in real-time applications. Finally, thermal limitations will give rise to new hotspot prediction and maximum power reduction techniques for real-time systems.

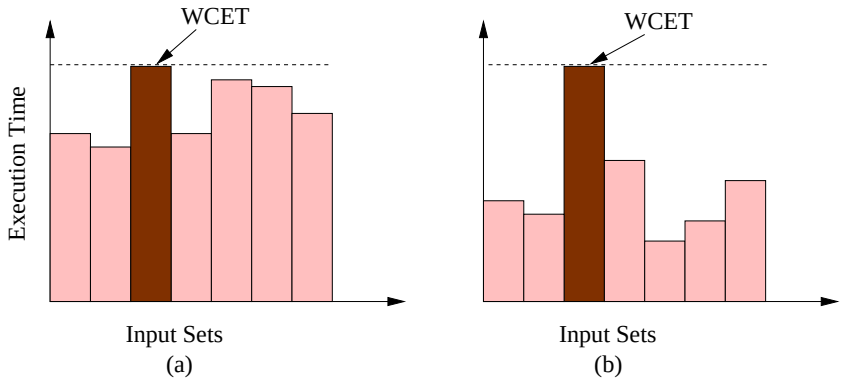


Fig. 8. Two applications with same WCET but with different suitability for DVS schemes.

ACKNOWLEDGMENT

The authors would like to thank Prof. C.M. Krishna for his many comments and suggestions which greatly improved this paper.

REFERENCES

[1] D. Singh, V. Tiwari, "Power Challenges in the Internet World", *Cool Chips Tutorial, In conjunction with the 32nd International Symposium on Microarchitecture, MICRO-32*, Nov. 1999, pp. 8-15.

[2] G. Johnson, "At Los Alamos, Two Visions of Supercomputing", *The New York Times*, June 25, 2002.

[3] http://www.handyscripts.co.uk/trubador_egg.htm

[4] T. Martin, D. Siewiorek, "A Power Metric for Mobile Systems", *International Symposium on Low Power Electronics and Design, ISLPED'96*, Aug. 1996, pp. 37-42.

[5] A. Wolfe, "Issues for Low-Power CAD Tools: A System Level Design Study", *Journal of Design Automation for Embedded Systems*, 1, 4, pp. 315-332, 1996.

[6] M. Pedram, Q. Wu, "Design Considerations for Battery-Powered Electronics", *36th Design Automation Conference, DAC99*, June 1999, pp. 861-866.

[7] K. Lahiri, A. Raghunathan, S. Dey, D. Panigrahi "Battery-Driven System Design: A New Frontier in Low Power Design", *Asia South Pacific Design Automation Conference (ASP-DAC) / International Conference on VLSI Design*, Jan. 2002, pp. 261-267.

[8] P. J. M. Havinga, G. J. M. Smit, "Energy-efficient Wireless Networking for Multimedia Applications", *Wireless Communications and Mobile Computing, Wiley*, 1, pp. 165-184, 2001.

[9] M. Pedram, "Power Optimization and Management in Embedded Systems", *Proceedings of the Conference on Asia South Pacific Design Automation Conference*, Jan. 2001, pp. 239-244.

[10] C. M. Krishna, K. Shin, *Real-Time Systems*, McGraw-Hill, 1997.

[11] J. A. Stankovic, K. Ramamritham (Eds.), *Tutorial: Hard Real-Time Systems*, IEEE Press, 1988.

[12] H. Kopetz, *Real-Time Systems Design Principles for Distributed Embedded Applications*, Kluwer Academic Publishers, Apr. 1997.

[13] C. Lee, J. K. Lee, T. Hwang, "Compiler Optimization on Instruction Scheduling for Low-Power", *Proceedings of The 13th International Symposium on System Synthesis*, Sep. 2000, pp. 55-60.

[14] I. Hong, M. M. Potkonjak, "Power Optimization Using Divide-and-Conquer Techniques for Minimization of the Number of Operations", *IEEE/ACM International Conference on Computer-Aided Design, ICCAD-97*, 1997, pp. 108-113.

[15] W. Fornaciari, M. Polentarutti, D. Sciuto, C. Silvano, "Power Optimization of System-level Address Buses Based on Software Profiling", *Proceedings of the Eighth International Workshop on Hardware/Software Codesign*, 2000, pp. 29-33.

[16] "Design: 2001 Edition", *Report of the International Technology Roadmap for Semiconductors*, 2001.

[17] Y.-K. Cheng, C.-H. Tsai, C.-C. Teng, S. Kang, *Electrothermal Analysis of VLSI Systems*, Kluwer Academic Publishers, 2000.

[18] A. Chandrakasan, R. Brodersen, *Low Power Digital CMOS Design*, Kluwer Academic Publishers, July 1995.

[19] F.N. Najm, "A Survey of Power Estimation Techniques in VLSI Circuits", *IEEE Transactions on VLSI Systems*, vol 2, no. 4, pp. 446-455, Dec. 1994.

[20] S. Devadas, S. Malik, "A Survey of Optimization Techniques Targeting Low Power VLSI Circuits", *32nd Design Automation Conference DAC95*, 1995, pp. 242-247.

[21] M. Pedram, "Power Minimization in IC Design: Principles and Applications", *ACM Transactions on Design Automation of Electronic Systems*, vol 1, no 1, Jan. 1996, pp. 3-56.

[22] D. M. Chapiro, "Globally Asynchronous Locally Synchronous Systems", *Ph.D. Thesis, Stanford University*, 1984.

[23] K. Wilcox, S. Manne, "Alpha Processors: A History of Power Issues and A Look to the Future", *CoolChips Tutorial, An Industrial Perspective on Low Power Processor Design, in Conjunction with 32nd International Symposium on Microarchitecture, MICRO-32*, pp. 16-37, 1999.

[24] F. Pollack, "New Microarchitecture Challenges in the Coming Generations of CMOS Process Technologies", *Keynote Talk, 32nd International Symposium on Microarchitecture, MICRO-32*, p. 2, Nov. 1999.

[25] L.Benini, G.De Micheli, "System-Level Power Optimization Techniques and Tools", *ACM Transactions on Design Automation for Embedded Systems (TODAES)*, Vol.5, No.2, pp. 115-192, Apr. 2000.

[26] E. Macii, M. Pedram, F. Somenzi, "High-Level Power Modeling, Estimation and Optimization", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol 17, no 11, pp. 1061-1079, Nov. 1998.

[27] R. Graybill, R. Melhem (Eds.), *Power Aware Computing*, Kluwer Academic/Plenum Publishers, ISBN 0-306-46786-0, May 2002.

[28] K. Roy, M.C. Johnson "Software Design for Low Power", *NATO Advanced Study Institute on Low Power Design in Deep Submicron Electronics*, Aug. 1996 (available at http://www.ece.purdue.edu/~ vlsi/papers/mark/lpsw_chap.ps).

[29] J. Lorch, A. J. Smith, "Software Strategies for Portable Computer Energy Management", *IEEE Personal Communications Magazine*, 5(3), pp. 60-73, June 1998.

[30] G. Welch, "A Survey of Power Management Techniques in Mobile Computing Operating Systems", *ACM Operating Systems Review (SIGOPS-OSR)*, Vol:29 No:4, pp.47-56, 1995.

[31] C. E. Jones, K. M. Sivalingam, P. Agrawal, J. C. Chen, "A Survey of Energy Efficient Network Protocols for Wireless Networks," *Wireless Networks*, vol 7, no 4, pp. 343-358, July 2001.

[32] N. Zhang, A. Burns, M. Nicholson, "Pipelined Processors and Worst-Case Execution Times", *Journal of Real-Time Systems*, Vol. 5, pp. 319-343, 1993.

[33] S.-T. Cheng, C.-M. Chen, J.-W. Hwang, "Low-Power Design for Real-Time Systems", *Real-Time Systems*, Vol. 15, pp. 131-148, Kluwer Academic Publishers, 1998.

[34] K. Danckaert, F. Catthoor, H. De Man, "System Level Memory Optimization for Hardware-Software Co-Design", *Proceedings of the 5th International Workshop on Hardware/Software Co-Design, CODES/CASHE'97*, 1997, pp. 55-64.

[35] L. Benini, A. Macii, E. Macii, M. Poncino, "Selective Instruction Compression for Memory Energy Reduction in Embedded Systems", *International Symposium on Low-Power Electronics and Design, ISLPED'99*, Aug. 1999, pp. 206-211.

[36] L. H. Lee, B. Moyer, J. Arends "Instruction Fetch Energy Reduction

Using Loop Caches for Embedded Applications with Small Tight Loops", *International Symposium on Low-Power Electronics and Design, ISLPED'99*, Aug. 1999, pp. 267-269.

[37] M. R. Stan, W. P. Burleson, "Bus-Invert Coding for Low-Power I/O", *IEEE Transactions on VLSI*, pp. 49-58, Mar. 1995.

[38] M. Mamidipaka, D. Hirschberg, N. Dutt, "Low Power Address Encoding Using Self-Organizing Lists," *International Symposium on Low-Power Electronics and Design, ISLPED'01*, 2001, pp. 188-193.

[39] Y. Li, W. Wolf, J. Henkel, "Task-level Memory Hierarchy Synthesis for Low Power in Real-Time Systems", *Proceedings of the 6th International Workshop on Hardware/Software Co-Design Codes/CASHE'98*, Mar. 1998 (available at http://www.ee.princeton.edu/~ yanbing/ftp/codes98.ps).

[40] C. L. Liu, J. WW. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real Time Environment", *Journal of the ACM*, Vol. 20, No. 1, pp. 46-61, 1973.

[41] F. Yao, A. Demers, S. Shenker, "A Scheduling Model for Reduced CPU Energy", *IEEE Annual Foundations of Computer Science*, pp. 374-382, 1995.

[42] D. Kirovski, M. Potkonjak, "System-Level Synthesis of Low-Power Hard Real-Time Systems", *Design Automation Conference, DAC'97*, 1997, pp. 697-702.

[43] S. Lee, T. Sakurai, "Run-Time Voltage Hopping for Low-Power Real-Time Systems," *37th ACM/IEEE Design Automation Conference, DAC'00*, June 2000, pp. 806-809.

[44] D.-I. Kang, S. Crago, J. Suh, "Power-Aware Design Synthesis Techniques for Distributed Real-Time Systems," *ACM Conference on Languages, Compilers, and Tools for Embedded Systems LCTES'01*, 2001, pp. 20-28.

[45] C. M. Krishna, Y-H. Lee, "Voltage-Clock-Scaling Adaptive Scheduling Techniques for Low Power in Hard Real-Time Systems," *Sixth IEEE Real Time Technology and Applications Symposium, RTAS'00*, June 2000, pp. 156-165.

[46] J. Luo, N. K. Jha, "Power-Conscious Joint Scheduling of Periodic Task Graphs and Aperiodic Tasks in Distributed Real-Time Embedded Systems", *International Conference on Computer-Aided Design*, Nov. 2000, pp. 357-364.

[47] D. Kirovski, P. Miodrag, "System-Level Synthesis of Low-Power Hard Real-Time Systems", *Computer Science Department Report, University of California, Los Angeles*, 1996.

[48] N. AbouGhazaleh, D. Moss, B. Childers, R. Melhem, "Toward The Placement of Power Management Points in Real Time Applications", *Workshop on Compilers and Operating Systems for Low Power, COLP'01*, pp. 3.1-3.7, 2001.

[49] I. Hong, G. Qu, M. Potkonjak, M. B. Srivastava, "Synthesis Techniques for Low-Power Hard Real-Time Systems on Variable Voltage Processors", *19th IEEE Real-Time Systems Symposium, RTSS'98*, 1998, pp. 178-187.

[50] T. Ishihara, H. Yasuura, "Voltage Scheduling Problem for Dynamically Variable Voltage Processors", *International Symposium on Low Power Electronics and Design, ISLPED'98*, 1998, pp. 197-202.

[51] Y. Shin, K. Choi, "Power Conscious Fixed Priority Scheduling for Hard Real-Time Systems", *36th ACM/IEEE Design Automation Conference, DAC'99*, 1999, pp. 134-139.

[52] A. Manzak, C. Chakrabarti, "Variable Voltage Task Scheduling for Minimizing Energy or Minimizing Power", *IEEE International Conference on Acoustics, Speech, and Signal Processing, (ICASSP'00)*, 2000, pp. 3239-3242.

[53] A. Manzak, C. Chakrabarti, "Variable Voltage Task Scheduling Algorithms for Minimizing Energy", *International Symposium on Low Power Electronics and Design, ISLPED'01*, 2001, pp. 279-282.

[54] J. Luo, N. K. Jha, "Battery-Aware Static Scheduling for Distributed Real-Time Embedded Systems", *38th ACM/IEEE Design Automation Conference, DAC'01*, 2001, pp. 444-449.

[55] J. Liu, P. H. Chou, N. Bagherzadeh, F. Kurdahi, "Power-Aware Scheduling under Timing Constraints for Mission-Critical Embedded Systems", *38th ACM/IEEE Design Automation Conference, DAC'01*, 2001, pp. 840-845.

[56] S.R. Park, W. Burleson, "Reconfiguration for Power Saving in Real-Time Motion-Estimation," *IEEE International Conference on Acoustics, Speech, and Signal Processing, (ICASSP'98)*, May 1998 (available at http://vsp2.ecs.umass.edu/vspg/publication/icassp98.ps.gz).

[57] J. Liu, P. H. Chou, N. Bagherzadeh, "Power-Aware Task Motion for Enhancing Dynamic Range of Embedded Systems with Renewable Energy Sources", *Workshop on Power-Aware Computer Systems (PACS'02), In conjunction with 8th Symposium on High-Performance Computer Architecture HPCA-8*, 2002 (available at http://www.ece.uci.edu/~ jinfeng/research/publication/pacs02.pdf).

[58] C. J. Hughes, J. Srinivasan, S. V. Adve, "Saving Energy with Architectural and Frequency Adaptations for Multimedia Applications", *34th International Symposium on Microarchitecture MICRO'01*, Dec. 2001, pp. 250-261.

[59] T. Martin, D. Siewiorek, "The Impact of Battery Capacity and Memory Bandwidth on CPU Speed-Setting: A Case Study", *International Symposium on Low Power Electronics and Design, ISLPED'99*, Aug. 1999, pp. 200-205.

[60] T. Martin, D. Siewiorek, "Non-Ideal Battery Behavior and Its Impact on Power Performance Trade-offs in Wearable Computing", *1999 International Symposium on Wearable Computers*, Oct. 1999, pp. 101-106.

[61] A. Azevedo, I. Issenin, R. Cornea, R. Gupta, N. Dutt, A. Veidenbaum, A. Nicolau, "Profile-based Dynamic Voltage Scheduling using Program Checkpoints in the COPPER Framework", *Design Automation and Test in Europe, DATE'02*, Mar. 2002, pp. 168-176.

[62] R. Levy, B. Crilly, B. Narahari, R. Simha, "Memory Issues in Power-aware Design of Embedded Systems: An Overview", *Second International Workshop on Compiler and Architecture Support for Embedded Systems, CASES'99*, 1999 (available at http://delta.cs.cinvestav.mx/~ pmejia/power/paper36.ps).

[63] T. Simunic, L. Benini, G. De Micheli, "Energy-efficient Design of Battery-Powered Embedded Systems", *International Symposium on Low Power Electronics and Design ISLPED'99*, 1999, pp. 212-217.

[64] P. Grun, N. Dutt, A. Nicolau "Access Pattern Based Local Memory Customization for Low Power Embedded Systems", *Design Automation and Test in Europe, DATE'01*, 2001, pp. 778-785.

[65] W.-T. Shiue, C. Chakrabarti, "Memory Exploration for Low Power Embedded Systems", *IEEE International Symposium on Circuit and Systems*, May 1999, pp. 250-253.

[66] C. Kulkarni, D. Moolenaar, L. Nachtergaele, F. Catthoor, H. De Man, "System-Level Energy-Delay Exploration for Multimedia Applications on Embedded Cores with Hardware Caches", *Journal of VLSI Signal Processing, Special Issue on SIPS'97, Workshop on Signal Processing Systems*, No.19, Kluwer, Boston, pp.45-58, 1999.

[67] C. Kulkarni, F. Catthoor, H. De Man, "Code Transformations for Low Power Caching in Embedded Multimedia Processors", *International Parallel and Distributed Processing Symposium (IPPS/SPDP 98)*, Apr. 1998, pp.292-297.

[68] D. Soudris, N. D. Zervas, A. Argyriou, M. Dasygenis, K. Tatas, C. Goutis, A. Thanailakis, "Data-Reuse and Parallel Embedded Architectures for Low-Power, Real-Time Multimedia Applications", *IEEE International Workshop on Power and Timing Modeling, Optimization and Simulation, PATMOS'00*, Sep. 2000, pp. 243- 254.

[69] O. S. Unsal, R. Ashok, I. Koren, C. M. Krishna, C. A. Moritz, "Cool-Cache for Hot Multimedia", *34th International Symposium on Microarchitecture MICRO34*, Dec. 2001, pp. 274-283.

[70] O. S. Unsal, I. Koren, C. M. Krishna, C. A. Moritz, "The Minimax Cache: An Energy-Efficient Framework for Media Processors", *8th International Symposium on High-Performance Computer Architecture, HPCA'02*, Feb. 2002, pp. 131-140.

[71] P. Pillai, K. G. Shin, "Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems", *18th ACM Symposium on Operating System Principles, SOSP'01*, Oct. 2001, pp. 89-102.

[72] V. Swaminathan, K. Chakrabarty, S.S. Iyengar, "Dynamic I/O Power Management for Hard Real-Time Systems", *International Symposium on Hardware/Software Co-Design CODES'01*, 2001, pp. 237-242.

[73] R. P. Dick, G. Lakshminarayana, A. Raghunathan, N. K. Jha, "Power Analysis of Embedded Operating Systems", *37th ACM/IEEE Design Automation Conference, DAC'00*, 2000, pp. 806-809.

[74] K. Baynes, C. Collins, E. Fiterman, C. Smit, T. Zhang, B. Jacob, "The Performance and Energy Consumption of Embedded Real-Time Operating Systems", *University of Maryland at College Park, Technical Report UMD-SCA-TR-2000-04*, Nov. 2000.

[75] A. Acquaviva, L. Benini, B. Ricco, "Energy Characterization of Embedded Real-Time Operating Systems", *Workshop on Compilers and Operating Systems for Low Power, COLP'01*, pp. 13-18, Sep. 2001 (available at http://research.ac.upc.es/pact01/colp/paper05.pdf).

[76] O. S. Unsal, I. Koren, C. M. Krishna, "High-Level Power-Reduction Heuristics in Large Scale Real-Time Systems", *IEEE International Workshop On Embedded Fault-Tolerant Systems*, Sep. 2000 (available at http://www.ecs.umass.edu/ece/realtime/publications/efts2000_umass.ps).

[77] K. I. Farkas, J. Flinn, G. Back, D. Grunwald, J. M. Anderson, "Quantifying the Energy Consumption of a Pocket Computer and a

Java Virtual Machine", *International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS 2000)*, June 2000, pp. 252-263.

[78] W. Yuan, K. Nahrsted, "A Middleware Framework Coordinating Processor/Power Resource Management for Multimedia Applications", *IEEE Symposium on Future Satellite Communications for Global IP and ATM Networking, Globecom 2001*, Nov. 2001, pp. 1984-1988.

[79] T. Ma, K. G. Shin, "A User-Customizable Energy-Adaptive Combined Static/Dynamic Scheduler for Mobile Applications", *Proceedings of IEEE Real-Time Systems Symposium, RTSS'00*, pp. 227-238, Nov. 2000 (available at http://kabru.eecs.umich.edu/papers/publications/2000/ma_rtss00.pdf).

[80] L. Benini, G. Castelli, A. Macii, R. Scarsi, "Battery-Driven Dynamic Power Management", *IEEE Design and Test of Computers*, Mar./Apr. 2001, pp. 53-60.

[81] J. Flinn, M. Satyanarayanan, "Energy-Aware Adaptation for Mobile Applications", *17th ACM Symposium on Operating System Principles, SOSP'99*, Dec. 1999, pp. 48-63.

[82] P. Bohrer, E. N. Elnozahy, T. Keller, M. Kistler, C. Lefurgy, R. Rajamony, "The Case for Power Management in Web Servers", *Power-Aware Computing, Kluwer/Plenum series in Computer Science*, Ch. 14, Jan. 2002.

[83] G. Qu, D. Kirovski, M. Potkonjak, M. B. Srivastava, "Energy Minimization of System Pipelines Using Multiple Voltages", *IEEE International Symposium on Circuits and Systems VLSI*, Vol.1, pp.362-365, 1999.

[84] L. Gruenwald, S. Banik, "A Power-Aware Technique to Manage Real-Time Database Transactions in Mobile Ad-Hoc Networks", *4th International Workshop on Mobility in Database and Distributed Systems*, pp. 570-574, Sep. 2001.

[85] O. S. Unsal, I. Koren, C. M. Krishna, "Power-Aware Replication of Data Structures in Distributed Embedded Real-Time Systems", *EHPC2000, 14th Annual International Parallel and Distributed Systems Symposium*, May 2000, pp. 839-846

[86] I. Lee, A. Philippou, O. Sokolsky, "Formal Modeling and Analysis of Power-Aware Real-Time Systems", *IEEE/IEE Workshop on Real-Time Embedded Systems, RTES'01*, Dec. 2001 (available at http://www.brics.dk/~ alcomft/TR/ALCOMFT-TR-02-137.ps.gz).

[87] D. Qiao, S. Choi, A. Soomro, K. G. Shin, "Energy-Efficient PCF Operation of IEEE 802.11a Wireless LAN", *IEEE Conference on Computer Communications INFOCOM'02*, pp. 580-589, June 2002 (available at http://kabru.eecs.umich.edu/papers/publications/2002/qiao_infocom02.pdf).

[88] K. Lahiri, A.Raghunathan, S.Dey, "Battery Efficient Architecture for an 802.11 MAC Processor", *International Conference on Communications, ICC'02*, pp. 669-674, May 2002.

[89] P. J. M. Havinga, G. J. M. Smit, "Minimizing energy consumption for handheld computers in Moby Dick", *23rd Euromicro Conference*, Sep. 1997, pp. 196-201.

[90] P. J. M. Havinga, G. J. M. Smit, "Octopus: embracing the energy efficiency of handheld multimedia computers", *ACM/IEEE 5th Annual International Conference on Mobile Computing and Networking, Mobicom99*, Aug. 1999, pp. 77-87.

[91] J. Gomez, A. T. Campbell, M. Naghshineh, C. Bisdikian, "Conserving Transmission Power in Wireless Ad Hoc Networks", *9th International Conference on Network Protocols, ICNP'01*, pp. 11-14, Nov. 2001.

[92] S. Chandra, "Wireless Network Interface Energy Consumption Implications of Popular Streaming Formats", *Multimedia Computing and Networking 2002, MMCN'02*, pp. 85-99, Jan. 2002.

[93] R. Min, M. Bhardwaj, S.-H. Cho, A. Sinha, E. Shih, A. Wang, A. Chandrakasan, "Low-Power Wireless Sensor Networks", *Fourteenth International Conference on VLSI Design*, Jan. 2001, pp. 205-210.

[94] J. Rabaey, J. Ammer, J. L. da Silva Jr., D. Patel, "PicoRadio: Ad-Hoc Wireless Networking of Ubiquitous Low-Energy Sensor/Monitor Nodes", *IEEE Computer Society Annual Workshop on VLSI (WVLSI'00)*, Apr., 2000, pp. 9-12.

[95] W. Heinzelman, A. Chandrakasan, H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," *Proceedings of the 33rd Hawaii International Conference on System Sciences (HICSS '00)*, Jan. 2000, pp. 3005-3014.

[96] W. Ye, J. Heidemann, D. Estrin, "An Energy-Efficient MAC protocol for Wireless Sensor Networks", *Proceedings of 21st Annual Joint Conference of the IEEE Computer and Communications Societies, (INFOCOM 2002)*, June, 2002, pp. 1567-1576.

[97] V. Tsiatsis, S. Zimbeck, M. Srivastava, "Architecture Strategies for Energy Efficient Packet Forwarding in Wireless Sensor Networks", *International Symposium on Low Power Electronics and Design, ISLPED'01*, 2001, pp. 92-95.

Osman S. Unsal received the B.S. degree in Computer and Control Engineering from Istanbul Technical University in Turkey, the M.S.E.E. from Brown University, Providence, RI and the Ph.D. in Electrical and Computer Engineering from University of Massachusetts, Amherst, MA in 1987, 1991 and 2003 respectively. Currently, he is with Intel Barcelona Research Center.

His research focus is in power-aware systems, computer architecture, compilers and real-time computing.

Israel Koren (S'72 - M'76 - SM'87 - F'91) received the B.Sc., M.Sc. and D.Sc. degrees from the Technion - Israel Institute of Technology, Haifa, in 1967, 1970, and 1975, respectively, all in Electrical Engineering. He is currently a Professor of Electrical and Computer Engineering at the University of Massachusetts, Amherst. Previously he was with the Technion - Israel Institute of Technology. He also held visiting positions with the University of California at Berkeley, University of Southern California, Los Angeles and University of California, Santa Barbara. He has been a consultant to several companies including IBM, Intel, Analog Devices, AMD, Digital Equipment Corp., National Semiconductor and Tolerant Systems.

Dr. Koren's current research interests include Techniques for Yield and Reliability Enhancement, Fault-Tolerant Architectures, Real-time systems and Computer Arithmetic. He published extensively in several IEEE Transactions and has over 170 publications in refereed journals and conferences. He currently serves on the Editorial Board of the IEEE Transactions on VLSI Systems. He was a Co-Guest Editor for the IEEE Transactions on Computers, special issue on High Yield VLSI Systems, April 1989 and the special issue on Computer Arithmetic, July 2000, and served on the Editorial Board of these Transactions between 1992 and 1997. He also served as General Chair, Program Chair and Program Committee member for numerous conferences. He has edited and co-authored the book, *Defect and Fault-Tolerance in VLSI Systems*, Vol. 1, Plenum, 1989. He is the author of the textbook *Computer Arithmetic Algorithms*, A.K. Peters, Ltd., 2002.