Low Power System Scheduling and Synthesis *

Niraj K. Jha
Department of Electrical Engineering
Princeton University
Princeton, NJ 08544

Abstract

Many scheduling techniques have been presented recently which exploit dynamic voltage scaling (DVS) and dynamic power management (DPM) for both uniprocessors and distributed systems, as well as both real-time and non-real-time systems. While such techniques are power-aware and aim at extending battery lifetimes for portable systems, they need to be augmented to make them battery-aware as well. We will survey such power-aware and battery-aware scheduling algorithms. Also, system synthesis algorithms for real-time systems-on-a-chip (SOCs), distributed and wireless client-server embedded systems, etc., have begun optimizing power consumption in addition to system price. We will survey such algorithms as well, and point out some open problems.

1. Introduction

Power consumption is well-recognized as one of the most important parameters in designing modern electronic systems. This is based on the need to increase the battery life of portable systems, reduce chip packaging and cooling costs, as well as reliability and environmental considerations. Since battery technology has not kept pace with the increasing power demands of such systems, it is important to target power during the design process. It is also known that higher the level of the design hierarchy where power is tackled, higher is the power reduction possible.

In this paper, we survey power reduction techniques that are applicable to the system level of the design hierarchy. We concentrate our attention on two specific areas: system scheduling and system synthesis. There are two techniques that can impact system scheduling: DVS and DPM. In DVS, different computation tasks are run at different voltages and clock frequencies in order to fill up the idle periods in the schedule, while still providing an adequate level of performance. DPM aims to shut off system parts not currently in use. An excellent survey for DPM can be found in [15]. We only cover DPM if it impacts system scheduling.

Low power system synthesis, specifically, SOC synthesis, and hardware-software co-synthesis of distributed embedded systems, has also attracted much attention. The system may be wireless and may have both quality of service (QoS) and real-time constraints.

We survey low power system scheduling and synthesis in Sections 2 and 3, respectively. We point out some open problems and conclude in Section 4.

2. Low Power System Scheduling

In this section, we survey various low power system scheduling techniques. Most target DVS, some DPM. Under DVS, many works target single processors, for both real-time and non-real-time applications. However, some recent work targets distributed systems.

Whenever both DVS and DPM are available for a processor, it is known that it is always advantageous to exploit DVS first.

The circuit delay depends on supply voltage V_{dd} as follows: $k \times V_{dd}/(V_{dd} - V_t)^2$, where k is a constant and V_t is the threshold voltage. Thus, delay increases as V_{dd} decreases. The switching power consumption (which is currently the dominant power consumption in CMOS technology) is given by $P = \alpha C_L V_{dd}^2 f$, where α is the switching activity, C_L is the load capacitance, and f is the frequency. $\alpha \times C_L$ is referred to as switched capacitance. Note that f is inversely proportional to circuit delay. Hence, $P \propto V_{dd}(V_{dd} - V_t)^2$.

The input specification of real-time distributed systems is frequently given in terms of a set of task graphs. A task graph is a directed acyclic graph in which a node is associated with a task and an edge is associated with the amount of data transferred between tasks. The period associated with a task graph indicates the time interval between its successive executions. A hard deadline, by which time the task must complete execution, is given for each sink node and some intermediate nodes. A multi-rate system consists of multiple task graphs with different periods. In addition to periodic task graphs, the system may also contain aperiodic tasks. An aperiodic task is invoked at any time and may have a hard or soft deadline. In case of soft deadlines, only the response time of the task needs to be minimized. For aperiodic tasks, generally a minimum inter-instance arrival time is specified.

For single processors, generally the tasks are assumed to be independent. The specification is assumed to be in the form of an arrival time, deadline and worst-case execution time for each task. Sometimes, the switched capacitance and period are also required.

Scheduling determines the time at which each task and communication executes.

2.1 DVS for uniprocessors

Initial work in this area was done for non-real-time systems. We survey these works first.

*Acknowledgment: This work was supported by DARPA under contract no. DAAB07-00-C-L516.

2.1.1 Non-real-time applications

In [1], DVS is used to maximize millions-of-instructions per Joule (MIPJ). This method is evaluated using trace-based simulation. The trace is divided into segments of various lengths. The run-times of segments are lengthened in order to eliminate idle time. In the previous segment, if the processor was more busy than idle, the speed is ramped up. If it was mostly idle, it is slowed down.

In [3], sophisticated heuristics are used to predict how busy the processor would be in the near future. This prediction is used to alter processor's speed and supply voltage. However, the smoothing heuristic given in [1] seems to perform as well as most of these techniques. Further evaluation of such DVS algorithms can be found in [6].

2.1.2 Real-time applications

A large body of DVS work exists for real-time applications running on single processors. In [2], a minimum-energy off-line preemptive scheduling algorithm is given. Suppose a set of tasks, J, needs to be executed in a given time interval. The *critical interval* for J is an interval in which a group of tasks must be scheduled at maximum, constant speed in any optimal schedule for J. The algorithm schedules this group of tasks, and constructs a subproblem for the remaining tasks and solves it recursively. This work also provides on-line scheduling extensions.

In [5], results are presented for processors in which the voltage can be continuously varied (e.g., Intel's XScale) and in which voltages can only be changed in discrete steps (e.g., Transmeta's Crusoe). In addition to task execution times and deadlines, the algorithm also requires the switched capacitance per cycle for each task. To minimize energy on a processor with continuously variable voltages, it shows that a unique voltage should be used for each task to expand its total execution time to its deadline. For processors with discrete voltages, at most two voltages need to be used to execute any task.

In [7], the problem of jointly scheduling periodic and aperiodic tasks is considered. The on-line algorithm guarantees all the deadlines of periodic tasks and tries to maximize the number of aperiodic tasks that can be accepted whose deadlines can also be met.

In [8], a low power fixed priority scheduling scheme exploits the fact that real-time tasks frequently run in less time than their specified worst-case execution time. When DVS is not applicable, it uses DPM. It takes into account the time taken to change the voltage and frequency. An extension of this method is given in [16]. The work in [23] also provides a low power fixed priority scheduler. It is based on the well-known rate monotonic scheduling algorithm.

In [13], the scheduling algorithm consists of an off-line phase in which voltages are chosen for tasks based on their worst-case execution times, and an on-line phase in which voltages are adjusted on-the-fly to reclaim any resource released by a task which finishes execution early. It is based on the traditional cyclic and earliest deadline first scheduling algorithms.

It is generally assumed that since energy is independent of the frequency of operation, unless frequency reduction is accompanied by voltage reduction, energy cannot be reduced. However, it has been found for real systems that operating at less than the maximum frequency can be advantageous. In [20], a theoretical explanation is given for this fact. It uses clock frequency reduction (not supply voltage) and DPM to reduce energy for multimedia applications.

The method in [26] assumes that when applications show bursty behavior, they must specify their future demands to the central scheduler. It proposes an energy priority scheduling heuristic which orders tasks based on the tightness of their deadlines and how often they overlap with other tasks. Low priority tasks are scheduled first since they can be easily preempted by higher priority tasks later.

In [25], the scheduling algorithm distributes the slack in a feasible schedule using an iterative algorithm which uses the fact that minimum energy is obtained when $\alpha(V_{dd}-V_t)^3$ is the same for all the tasks for both periodic and aperiodic schedules.

All the above methods apply DVS at the task level. However, the work in [24] shows that further energy reduction can be obtained if intra-task scheduling is done. It uses static timing analysis of the task to control the voltage within the task boundary. It partitions the task into several segments, each assigned with a separate voltage. In [14] as well, the idea of intra-task scheduling is exploited by partitioning each task into fixed-length timeslots. However, unlike [24], it does not provide guidelines for selecting the best program locations where voltage scaling code can be inserted.

2.2 DVS for distributed systems

DVS techniques for distributed systems, which consist of multiple processing elements (PEs) connected with a communication network, are discussed next.

2.2.1 Power-aware techniques

In [17], a power-conscious algorithm is given for jointly scheduling multi-rate periodic task graphs with hard deadlines and aperiodic tasks with hard or soft deadlines. Periodic tasks are first scheduled statically and room made in the schedule for hard aperiodic tasks. Soft aperiodic tasks are scheduled dynamically with an on-line scheduler. It exploits the concepts of slack stealing and resource reclaiming to minimize the response times of aperiodic tasks. It uses DPM for parts of the schedule where DVS is not applicable.

In [18], a list scheduling technique chooses the best two supply voltages for each task in a task graph specification. It uses dynamic recalculation of energy-sensitive task priorities for this purpose.

In [19], a hybrid global/local search optimization framework is given for DVS. Performance is a constraint under which an attempt is made to find the optimum voltage level for all the tasks that need to be executed. The schedule of tasks on different processors is assumed to be known *a priori*. The power consumed by the DVS hardware and the time to switch between voltages are also taken into account. A genetic algorithm is used for global search (coupled with a technique called simulated heating) and hill climbing and Monte Carlo techniques for local search.

In [21], a power-aware scheduling algorithm is presented for mission-critical applications. It satisfies min/max timing constraints and max power constraint. In addition, it also tries to meet min power constraints in order to make full use of free power (e.g., solar power)

or to control power jitter.

2.2.2 Battery-aware techniques

In [22], two battery-aware static scheduling techniques are presented. As suggested in [9, 10], reducing the discharge current level and shaping its distribution are essential for extending battery lifespan. This is based on the observation that battery capacity decreases as the discharge current increases. The first scheduling technique in [22] optimizes the discharge power profile in order to maximize the utilization of battery capacity. The second technique efficiently re-allocates slack time to better enable DVS. This helps reduce the average discharge power consumption as well as flatten the discharge power profile.

2.3 DPM-based system scheduling

In [4], tasks from task graphs are assumed to be assigned to processors in a distributed system. The scheduling algorithm reduces the delay/energy overheads for changes between sleep and wakeup states by inserting buffers between processors and rescheduling.

A similar idea is used in [12], where task execution is ordered to adjust the length of the idle periods in the schedule, thus enabling better DPM. Since putting a processor to sleep is worth it only if the sleep time is long enough to compensate for the delay/energy overheads of DPM, the idea behind this method is to make idle periods clustered and long.

3. Low Power System Synthesis

In this section, we survey various system synthesis techniques for low power SOCs and distributed embedded systems.

The key steps in system synthesis are *allocation*, *assignment*, *scheduling*, and *performance evaluation*. Allocation determines the number of each type of PE and communication link in the system architecture. Assignment chooses a PE (link) to execute each task (communication) upon. As mentioned earlier, scheduling determines the time of execution of each task and communication. Performance evaluation involves computing the price, speed and power of the system architecture.

3.1 Low power SOC synthesis

The method in [28] considers an SOC with a fixed allocation of one processor, ASIC, instruction cache, data cache and main memory. As a case study, an MPEG-2 encoder is chosen to investigate the impact of different hardware/software partitions of the input specification between the processor and ASIC, and different system configurations such as cache size, cache line size, cache associativity and main memory size, on power dissipation of the SOC. The extension of this method in [29] assumes the same SOC architecture, however, with the hardware fixed. The software is changed through various high-level transformations. This impacts cache and memory parameters. It investigates overall system energy. Another extension of this approach is given in [34].

In [32], the allocation of the SOC architecture is not fixed beforehand, and is hence not limited to a single processor and ASIC. It describes a tool called MOCSYN which synthesizes real-time heterogeneous single-chip hardware/software architectures using an adaptive multi-objective genetic algorithm. It starts with a system specification consisting of multiple periodic task graphs as well as a database of core and SOC characteristics. The database consists of the worst-case execution times and average/peak power consumption of each task on each core on which the task can possibly run. Each core has a width, height, maximum clock frequency, variable indicating whether or not its communication is buffered, and energy consumption per cycle dedicated to communication. In addition, information on core price is also available. A single system synthesis run produces multiple SOC designs which trade off system price, power and area under real-time constraints. It assumes asynchronous communication between synchronous cores on the SOC and determines the best clock frequency for each core. It produces a hierarchical bus structure which balances ease of layout with the reduction of bus contention. It also performs floorplanning in the inner loop of system synthesis to accurately estimate global communication delays and energy, as well as clock network energy.

In [35], the target SOC architecture consists of one processor, instruction/data cache, main memory, and several ASICs and peripherals. It shows the importance of adequate adaptation between core and interface parameters to minimize power consumption. Cache parameters and configurations of cache buses have a significant effect in this respect.

In [36], DVS is integrated into SOC synthesis. The target architecture consists of one processor and instruction/data cache. The input specification is assumed to consist of independent periodic tasks. The technique also addresses selection of the best processor core and determination of cache size and configuration to best enable DVS.

3.2 Low power distributed system synthesis

In a distributed system, PEs are not limited to a single chip. However, system synthesis still consists of solving the allocation, assignment, scheduling and performance evaluation problems. We discuss various types of distributed system synthesis algorithms next.

3.2.1 Iterative improvement algorithm

In [27], the first work to integrate DVS into system synthesis is presented. Independent periodic tasks are assumed which are mapped to multiple processors connected by a bus. In addition to voltage, the switched capacitance is also implicitly optimized. Resource allocation is done using a gradient-driven iterative improvement heuristic. Tasks are also iteratively assigned to allocated processors based on an objective function. Additional load balancing is attempted in a post-processing step.

3.2.2 Constructive algorithms

In [33], a constructive algorithm called COSYN is described which starts with a set of multi-rate periodic task graphs with real-time constraints and produces a price and power optimized distributed system architecture. It uses a combination of preemptive and non-preemptive scheduling. However, use of preemptive scheduling should be avoided as much as possible since it increases power consumption. It performs task clustering before system synthesis to make synthesis more tractable. This allows it to tackle very large task graphs (with more than a thousand tasks). It uses the concept of task graph pipelining to handle task graphs in which period is less than the deadline.

For medium- to large-scale embedded systems, such as telecom transport systems, task graphs are usually hierarchical, i.e., each node in an upper-level task graph may correspond to a full-fledged task graph at a lower level. If flat, non-hierarchical system architectures are derived from such hierarchical task graphs, many communication and processing bottlenecks may be created. In [30], a constructive algorithm called COHRA is given to synthesize hierarchical distributed architectures from hierarchical or non-hierarchical real-time periodic task graphs. A hierarchical architecture is obtained by composing lower-level sub-architectures. COHRA also optimizes power consumption and fault tolerance.

3.2.3 Genetic algorithms

In [31], a genetic algorithm called MOGAC is used to synthesize real-time heterogeneous distributed architectures from multi-rate real-time periodic task graph specifications. It optimizes both price and power. Genetic algorithms excel at such multiobjective optimization. The number and type of PEs are not fixed *a priori*. Genetic algorithms allow solutions to be cooperatively share information with each other, exploring the set of solutions that can only be improved in one way by being degraded in another (the *Pareto-optimal* set). MOGAC uses heuristics to allow multi-rate systems to be scheduled in reasonable time even when the periods are very different and possibly co-prime.

In [37], a genetic algorithm called COWLS targets embedded systems consisting of servers and low-power clients which communicate with each other through a channel of limited bandwidth, e.g., a wireless link. Clients may be mobile. It simultaneously optimizes the price of the client-server system, power consumption of the client, and response times of tasks with only soft deadlines, while meeting all the hard deadlines. It produces numerous solutions which trade off architectural features such as price, power and response time.

A genetic algorithm is also used in [39] to incorporate DVS into an energy minimization technique for distributed embedded systems. It takes the power variations of tasks into account while performing DVS. An off-line voltage scaling heuristic is proposed which is fast enough to be used in system synthesis, starting from real-time periodic task graphs.

3.2.4 QoS driven system synthesis

QoS is an important consideration in designing systems for real-time multimedia and wireless communication applications. In [38], a DVS technique for partitioning a set of applications among multiple processors is given which minimizes system energy while satisfying individual QoS requirements. QoS is a function of the required resources, such as bandwidth, CPU time, and buffer space. The applications are assumed to be independent, have the same arrival times and no deadline constraints.

4. Conclusions and Open Problems

From the above discussions, one may conclude that the field of DVS for uniprocessors for both real-time and non-real-time applications has attained sufficient maturity. That cannot be said about low power system scheduling for distributed systems. Given that when both are applicable to processors in a distributed system, DVS is better than DPM, purely DPM-based system scheduling may not be that useful an area to pursue. This is specially true because many processors in the future are likely to have DVS capability. Of course, DPM will continue to be useful for other parts of the system which do not have DVS capability. Also, DVS cannot always get rid of all the idle slots in the system schedule. Thus, a combined DVS+DPM approach is preferable, applying DVS before DPM.

Battery-aware DVS+DPM approaches need further investigation. It is known that reducing the workload in a battery-operated system for a period of time leads to a recovery effect, which results in an increase in battery capacity [11]. This effect needs to be exploited in system scheduling.

Low power system synthesis is also not a mature area. Currently, most such algorithms assume that the average power consumption of each task on each type of PE it can run on has been given. This is done because using more sophisticated power estimation techniques for processors, FPGAs and ASICs in the inner loop of system synthesis is currently not feasible. This points to the need for fast, yet relatively accurate, power estimation techniques, such as high-level macromodels, to drive system synthesis. Although some progress has been made in this direction for ASICs, high-level energy or power macromodels for processors and FPGAs are rare [40, 41]. After obtaining such macromodels, the next step would be to integrate them in the inner loop of system synthesis.

Finally, when more sophisticated power-aware and battery-aware distributed system scheduling algorithms become available, they also need to be integrated in the inner loop of system synthesis.

References

[1] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for reduced CPU energy," in *Proc. Symp. Operating Systems Design & Implementation*, Usenix Association, Nov. 1994.

[2] F. Yao, A. Demers, and S. Shenker, "A scheduling model for reduced CPU energy," in *Proc. Ann. Symp. Foundations Computer Science*, pp. 374-381, 1995.

[3] K. Govil, E. Chan, and H. Wasserman, "Comparing algorithms for dynamic speed-setting of a low-power CPU," in *Proc. MOBICOM*, pp. 13-25, 1995.

[4] J. J. Brown, D. Z. Chen, G. W. Greenwood, X. Hu, and R. W. Taylor, "Scheduling for power reduction in a real-time system," in *Proc. Int. Symp. Low Power Electronics & Design*, pp. 84-87, Aug. 1997.

[5] T. Ishihara and H. Yasuura, "Voltage scheduling problem for dynamically variable voltage processors," in *Proc. Int. Symp. Low Power Electronics & Design*, pp. 197-202, Aug. 1998.

[6] T. Pering, T. Burd, and R. Brodersen, "The simulation and evaluation of dynamic voltage scaling algorithms," in *Proc. Int. Symp. Low Power Electronics & Design*, pp. 76-81, Aug. 1998.

[7] I. Hong, M. Potkonjak, and M. B. Srivastava, "On-line scheduling of hard real-time tasks on variable voltage processor," in *Proc. Int. Conf Computer-Aided Design*, pp. 653-656, Nov. 1998.

[8] Y. Shin and K. Choi, "Power conscious fixed priority scheduling for hard real-time systems," in *Proc. Design Automation Conf.*, pp. 134-139, June 1999.

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

[10] T. Martin and D. Siewiorek, "The impact of battery capacity and memory bandwidth on CPU speed-setting: A case study," in *Proc. Int. Symp. Low Power Electronics & Design*, pp. 200-205, Aug. 1999.

[11] D. Linden, *Handbook of Batteries and Fuel Cells*, McGraw-Hill, NY, 1984.

[12] Y.-H. Lu, L. Benini, and G. De Micheli, "Low-power task scheduling for multiple devices," in *Proc. Int. Wkshp. HW/SW Co-design*, pp. 39-43, Mar. 2000.

[13] C. M. Krishna and Y.-H. Lee, "Voltage-clock-scaling adaptive scheduling techniques for low power in hard real-time systems," in *Proc. Real-Time Technology & Applications Symp.*, May 2000.

[14] S. Lee and T. Sakurai, "Run-time voltage hopping for low-power real-time systems," in *Proc. Design Automation Conf.*, pp. 806-809, June 2000.

[15] L. Benini, A. Bogliolo, and G. De Micheli, "A survey of design techniques for system-level dynamic power management," *IEEE Trans. VLSI Systems*, vol. 8, no. 3, pp. 299-316, June 2000.

[16] Y. Shin, K. Choi, and T. Sakurai, "Power optimization of real-time embedded systems on variable speed processors," in *Proc. Int. Conf. Computer-Aided Design*, pp. 365-368, Nov. 2000.

[17] J. Luo and N. K. Jha, "Power-conscious joint scheduling of periodic task graphs and aperiodic tasks in distributed real-time embedded systems," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 2000.

[18] F. Gruian and K. Kuchcinski, "LEneS: Task scheduling for low-energy systems using variable supply voltage processors," in *Proc. Asia South Pacific Design Automation Conf.*, pp. 449-455, Jan. 2001.

[19] N. K. Bambha, S. S. Bhattacharya, J. Teich, and E. Zitzler, "Hybrid global/local search strategies for dynamic voltage scaling in embedded multiprocessors," in *Proc. Int. Wkshp. HW/SW Co-design*, pp. 243-248, Mar. 2001.

[20] A. Acquaviva, L. Benini, and B. Ricco, "An adaptive algorithm for low power streaming multimedia processing," in *Proc. Design Automat. & Test in Europe Conf.*, Mar. 2001.

[21] J. Liu, P. H. Chou, N. Bagherzadeh, and F. Kurdahi, "Power-aware scheduling under timing constraints for mission-critical embedded systems," in *Proc. Design Automation Conf.*, pp. 840-845, June 2001.

[22] 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.

[23] G. Quan and X. Hu, "Energy efficient fixed-priority scheduling for real-time systems on variable voltage processors," in *Proc. Design Automation Conf.*, pp. 828-833, June 2001.

[24] D. Shin, J. Kim, and S. Lee, "Low-energy intra-task voltage scheduling using static timing analysis," in *Proc. Design Automation Conf.*, pp. 438-443, June 2001.

[25] A. Manzak and C. Chakrabarti, "Variable voltage task scheduling algorithms for minimizing energy," in *Proc. Int. Symp. Low Power Electronics & Design*, Aug. 2001.

[26] J. Pouwelse, K. Langendoen, and H. Sips, "Energy priority scheduling for variable voltage processors," in *Proc. Int. Symp. Low Power Electronics & Design*, Aug. 2001.

[27] D. Kirovski and M. Potkonjak, "System-level synthesis of low power hard real-time systems," in *Proc. Design Automation Conf.*, pp. 697-702, June 1997.

[28] J. Henkel and Y. Li, "Energy-conscious HW/SW-partitioning of embedded systems: A case study of an MPEG-2 encoder," in *Proc. Int. Wkshp. HW/SW Co-design*, pp. 23-27, Mar. 1998.

[29] Y. Li and J. Henkel, "A framework for estimating and minimizing energy dissipation of embedded HW/SW systems," in *Proc. Design Automation Conf.*, pp. 188-193, June 1998.

[30] B. Dave and N. K. Jha, "COHRA: Hardware-software co-synthesis of hierarchical heterogeneous distributed embedded systems," *IEEE Trans. Computer-Aided Design*, vol. 17, Oct. 1998.

[31] R. P. Dick and N. K. Jha, "MOGAC: A multiobjective genetic algorithm for the hardware-software co-synthesis of distributed embedded systems," *IEEE Trans. Computer-Aided Design*, vol. 17, Oct. 1998.

[32] R. P. Dick and N. K. Jha, "MOCSYN: Multiobjective core-based single-chip system synthesis," in *Proc. Design Automat. & Test in Europe Conf.*, Feb. 1999.

[33] B. Dave, G. Lakshminarayana, and N. K. Jha, "COSYN: Hardware-software co-synthesis of embedded systems," *IEEE Trans. VLSI Systems*, vol. 7, Mar. 1999.

[34] J. Henkel, "A low power hardware/software partitioning approach for core-based embedded systems," in *Proc. Design Automation Conf.*, pp. 122-127, June 1999.

[35] T. D. Givargis, J. Henkel, and F. Vahid, "Interface and cache power exploration for core-based embedded system design," in *Proc. Int. Conf. Computer-Aided Design*, pp. 270-273, Nov. 1999.

[36] I. Hong, D. Kirovski, G. Qu, M. Potkonjak, and M. B. Srivastava, "Power optimization of variable voltage core-based systems," *IEEE Trans. Computer-Aided Design*, vol. 18, no. 12, pp. 1702-1714, Dec. 1999.

[37] R. P. Dick and N. K. Jha, "COWLS: Hardware-software co-synthesis of distributed wireless low-power embedded client-server systems," in *Proc. Int. Conf. VLSI Design*, Jan. 2000.

[38] G. Qu and M. Potkonjak, "Energy minimization with quality of service," in *Proc. Int. Symp. Low Power Electronics & Design*, pp. 43-49, Aug. 2000.

[39] M. Schmitz and B. M. Al-Hashimi, "Considering power variations of DVS processing elements for energy minimization in distributed systems," in *Proc. Int. Symp. System Synthesis*, 2001.

[40] T. K. Tan, A. Raghunathan, G. Lakshminarayana, and N. K. Jha, "High-level software energy macro-modeling," in *Proc. Design Automation Conf.*, pp. 605-610, June 2001.

[41] L. Shang and N. K. Jha, "High-level power modeling of CPLDs and FPGAs," in *Proc. Int. Conf. Computer Design*, Sept. 2001.