A Simple Power-Aware Scheduling for Multicore Systems when Running Real-Time Applications

Diana Bautista, Julio Sahuquillo, Houcine Hassan, Salvador Petit, José Duato
Department of Computer Engineering (DISCA)
Universidad Politécnica de Valencia, Spain
diabaura@fiv.upv.es, {jsahuqui,husein,spetit}@disca.upv.es

Abstract

High-performance microprocessors, e.g., multithreaded and multicore processors, are being implemented in embedded real-time systems because of the increasing computational requirements. These complex microprocessors have two major drawbacks when they are used for real-time purposes. First, their complexity difficults the calculation of the WCET (Worst Case Execution Time). Second, power consumption requirements are much larger, which is a major concern in these systems.

In this paper we propose a novel soft power-aware real-time scheduler for a state-of-the-art multicore multithreaded processor, which implements dynamic voltage scaling techniques. The proposed scheduler reduces the energy consumption while satisfying the constraints of soft real-time applications. Different scheduling alternatives have been evaluated, and experimental results show that using a fair scheduling policy, the proposed algorithm provides, on average, energy savings ranging from 34% to 74%.

1 Introduction

Embedded systems are experiencing a growth of the number of functionalities they incorporate. Among others, they can act as phone cells, PDA's, car onboard systems, etc. Due to the performance requirements of these applications, high-end embedded processors nowadays are including complex mechanisms developed for high-performance architectures, such as pipelining, dynamic prediction, out-of-order execution, or dynamic multithreading. For example, the Ubicom IP3023 [28] implements 8 hardware threads, while the ARM11 [19] incorporates an 8-stage pipeline, caches, and dynamic branch prediction. Moreover, multicore implementations of the ARM11 have been performed [13].

Unfortunately, these high performance techniques difficult the calculation of the WCET (Worst Case Execution Time) of real-time tasks. A major concern is to guarantee the deadline constraint of a real-time task executed by a thread when it has to concurrently run with other threads [11, 2]. A simple solution, like running the real-time tasks alone to guarantee predictability, can degrade dramatically the overall performance [5]. To deal with these aspects, the scheduler should be able to control more precisely the sharing of processor's internal resources by real-time tasks.

On the other hand, the application of high-performance techniques in embedded processors require, as an important side effect, much larger amounts of energy. Therefore, research on energy-aware techniques for embedded processors has been gaining significance [14, 26, 29].

In this context, dynamic voltage scaling (DVS) [14] is a widely implemented techniques for energy efficiency. For instance, the Transmeta Crusoe [8], the Intel Xeon [7] and the Mobile AMD DuronTM [9] implement DVS. This mechanism entails reducing the system energy consumption by reducing the CPU supply voltage and the clock frequency simultaneously. However, the provided energy gains increase the thread execution times, so creating conditions where the system is overloaded that can jeopardize the schedulability of real-time tasks [1, 3, 20].

Due to the reasons described above, new real-time schedulers addressing multithreaded and multicore embedded environments are required. These schedulers must aim to keep low the energy consumption while adapting the processor speed to the requirements of the real-time loads. In other words, scheduler must the trade-off real-time constraints versus energy savings.

This paper focuses on design issues of a real-time power-aware scheduler for a high-performance embedded processor (i.e, multithreaded and multicore). This scheduler adapts the global frequency of the cores to the computation requirements of soft real-time tasks while improving energy savings. The scheduler pursues to minimize the number of DVS transitions by increasing or decreasing the voltage and frequency of all the cores at the same time. Frequency increases when it is required to satisfy the time requirements of real-time tasks. On the the other hand, when a task finishes its execution, if its computation time can be guaranteed, the frequency is reduced.

The proposed scheduler has been evaluated on a model of a current high-end ARM embedded microprocessor (used in several of devices, like Nokia N95, HTC TN II , Qualcomm MSM7200, Sony Ericsson W series, Nintendo DS, iPod) executing a set of standard benchmarks of the EEMB [6] and the MediaBench suites [18] and the WCET analysis project Benchmark [24].

The remaining of this paper is structured as follows. Section 2 discusses important research on the topic of energy management of real-time systems. Section 3 describes the proposed scheduling algorithm and the modelled system. Section 4 evaluates the algorithm and analyzes the experimental results. Finally, section 5 presents some concluding remarks.

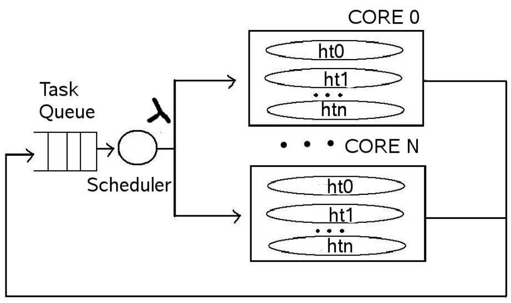


Figure 1. Modelled task queue and multicore system.

2 Related Work

There are three main models used to implement multithreading capabilities in current commercial processors: fine grain multithreading, coarse grain multithreading and simultaneous multithreading [25]. For example Intel's Montecito [21] implements coarse grain multithreading while the Sun's Niagara multicore architecture [16] presents one fine grain multithreaded pipeline per core. Finally, several simultaneous multithreading implementations exist in the market [12, 15].

Although, in general, fine grain and simultaneous multithreading approaches offer more performance by allowing dynamic sharing of pipeline resources in a cycle-by-cycle basis, guaranteeing a-priori the WCET of real-time tasks sharing the processor resources under these models becomes extremely difficult. Therefore, in this paper, we assume a coarse grain model, which switches threads on long latency events, as classical multitask operating systems do.

Despite the large amount of research focussing on multithreading, to the best of our knowledge, no power-aware scheduling algorithm has been published dealing with both coarse grain multithreaded processors and real-time constraints.

A large amount of research papers have explored energy management on uniprocessor real-time systems. Most of them focus on periodic task systems [20, 31], but also some proposals have been published dealing with aperiodic tasks [23] and sporadic tasks [22]. Buttazzo et al. [20] present an algorithm for energy management based on DVS that integrates the elastic scheduler for discrete voltage mode processors.

Some research studies have addressed the problem of energy management in multiprocessor platforms when running real-time tasks [4, 1, 3]. In [4] the problem of reducing energy for periodic real-time tasks in symmetric multiprocessors using DVS is presented. The EDF (Earliest Deadline First) algorithm analyzes the effect of partitioning heuristics. In [1], Al Enawy et al. consider the problem of energy minimization for periodic preemptive hard real-time tasks that are scheduled on a symmetric multiprocessor platform with DVS capability. They adopt partitioned scheduling and assume that tasks are assigned static rate-monotonic priorities. In [3], Baruah addresses the system synthesis problem of periodic real-time tasks on identical multiprocessors using global scheduling with EDF.

Some research has also focused on multithreaded processors. In [5] Cazorla et al. present an architecture where an SMT interacts with the OS to improve performance. The OS specifies the predictable performance threads (PPT) to be executed at a given frequency while the non-PPT use the resources that the PPT do not need. However, the problem of combining energy management with the schedulability guarantee

```
Initial state: both cores are switched off

Filling cores: applies when there is a ready task T

Step1: Is there any core empty?
              If so, if core A is empty then launch T to core A,
                     otherwise launch T to core B
              If not, go to step 2.

Step2: Launch T to the core less loaded,
              If both cores are equally loaded then
                   increase frequency and launch T to core A.

Reducing frequency: applies when a task T finishes

Step3: Are both cores equally loaded?
       If so, reduce frequency
```

Table 1. Scheduling algorithm.

of real-time tasks is not resolved. Moreover in [11] Rotenberg states that SMT processors difficult proving hard-real-time guarantees, and proposes a virtual in-order superscalar architecture that allows the virtualization of a superscalar in various processors. This architecture incorporates a static scheduler to execute periodic tasks. They compare their scheduler to the EDF for multiprocessors in terms of rate of deadline misses. Therefore, the schedulability of hard real-time tasks is not guaranteed and the energy management is not tackled.

3 System Architecture and Scheduling Algorithm

The system modelled in this work consists of a coarse grain multithreaded multicore system with a task queue feeding the scheduler as shown in Figure 1. Although the results can be extended to any number of cores, for evaluation purposes (see Section 4.3) we consider a bi-core.

This scheduler works as follows. When a task finishes its execution it comes back to a global task queue structure. This queue is read by the scheduler, which launch them to the corresponding core. Soft aperiodic tasks are characterized by the minimum inter-arrival time parameter. This parameter is modeled by λ. Tasks are launched at a λ rate, i.e., the inter-arrival time is $1/\ \lambda$ *units of time*. This time refers to the time that the scheduler waits between the submission of two consecutive tasks to the system. It has been chosen three values of lambda to analyze its effect on the consumption of energy. For evaluation purposes it has been assumed the stand-alone execution time of the slowest task in the mix as the *baseline* inter-arrival time. However, as the processor effectiveness will depend on the inter-arrival time, different scenarios have been analyzed. To this end, the baseline inter-arrival time has been multiplied by a given factor K (i.e., 1, 1.5, and 2).

The proposed algorithm aims at minimizing the energy consumption while ensuring that the WCET of the running tasks is fulfilled. We have assumed a sim-

Table 2. Machine Parameters.

Microprocessor core	
Issue policy	In order
Fetch kind	Switch on event
Branch Prediction	Two-level global history
	256 entries BTB, 4096 2-bit
	saturating counters GHB
	13 cycles misprediction
Issue bandwidth	2 instructions/cycle
# of Integer ALU's, multiplier/dividers	2,1
# of FP ALU's, multiplier/dividers	2,1
Memory Hierarchy	
Cache memory	Disabled
Memory access latency	100 cycles

ple model where one core working at its lowest speed can guarantees the WCET of one only task running in it with low speed. If two tasks are being executed in the same core, to guarantee their WCET the core must run at medium speed, and if the number of tasks running in the same core is three or more, the speed must be the highest one. Static schedulability analysis of the system has been performed off-line to state the previous hypothesis. A static schedulability analysis for all the situations arising in the system has been performed during the design phase. This analysis has taken into account the execution times and deadlines of the benchmarks and the inter-arrival rate parameter.

In the initial state, both cores are switched off, so consuming no power. Then, when a task is ready to run,one core is switched on and starts working at its minimum speed. Then, if another task is ready, the scheduler estimates if that core satisfies the WCET of both of the tasks, (i.e., the running one and the incoming one), at the current speed. If this is not possible, the second core is also switched with the minimum speed. From this point, if we cannot guarantee the WCET of additional incoming tasks we increase the frequency up to the maximum one. As tasks finalize this process is reversed, reducing the frequency and shutting down the cores on an individual basis. Table 1 shows a detailed description of the Scheduling Algorithm.

The proposed algorithm globally increases and decreases the voltage and frequency of both processors at the same time. Although it is possible to use different voltages and frequencies in each processor (i.e, per-core DVS), this option is more complex and expensive, since it requires more voltage regulators and complicates the power delivery network. In addition, it has been shown [10] that global DVS can have an efficiency very close to per-core DVS if the load is fairy balanced.

Table 3. Description of Benchmarks.

Name	Description
Adpcm	Speech compression and decompression algorithm
Crc	Cyclic redundancy check computation on 40 bytes of data.
Fft1	1024-point Fast Fourier Transform using the Cooly-Turkey algorithm.
Fir	Finite impulse response filter over a 700 items long sample.
Lucmp	LU decomposition algorithm
Fdct	Fast Discrete Cosine Transform.
Mpeg	Lossymotion video compression decoder
G721	Voice compression coder based on G.711, G.721 and G.723 standards (encode,decode)
Rawc	Speech compression algorithm

Table 4. Benchmarks characteristics

Tasks	Execution time (K)	Memory inst.(%)	CPU time(%)	Overlap (%)
crc	171.0	21.6	59.5	2.0
fft1	47.8	24.5	63.0	10.2
ludcmp	57.5	24.5	61.4	7.4
fdct	50.1	24.8	59.6	5.4
fir	51.1	23.6	61.2	6.3
rawc	9400	6.4	88.8	4.7
adpcm	1008.2	23.9	64.1	3.5
mpeg	18900	22.1	72.4	19.2
encode	21800	26.5	56.7	5.2
decode	21900	26.6	56.6	5.0

4 Experimental Results

The proposed techniques have been evaluated on top of the Multi2Sim simulation framework [27] which was extended to implement a scheduler (i.e., the proposed algorithm) and to support the input of tasks in an aperiodic manner (softrealtime).

This section evaluates a bi-core multithreaded system, where the microarchitecture of each core resembles the embedded ARM 11 microprocessor. Table 2 summarizes the architectural parameters. Experimental results were obtained by using mixes of applications from the Mediabench and EEMBC. Table 3 shows a description of these benchmarks.

4.1 Workload characterization

Since the characteristics of the load can have influence in the energy consumed, before designing mixes it is important to explore the execution requirements of each benchmark in the underlying system. To this end, this section analyzes for each benchmark, its computation-memory requirements as well as the potential overlapping among these resources.

Table 4 shows the results obtained for the EEMBC and Mediabench benchmarks suites respectively. As observed, the EEMBC benchmarks are much shorter (e.g, on the average, several orders of magnitude) than the Mediabench ones. On the other hand, the percentage of memory instruction in between 21 and 27% across all benchmarks falls, with the exception of `rawd` and `rawc` from Mediabench where that percentage is below 7%. The memory-CPU overlapping has been obtained attending to the percentage of time that the processor spends with memory or CPU and comparing it with the total execution time quantified in processor cycles. Looking at this result, we can see that in some applications this value is quite low despite their percentage of memory instructions, since only in `mpeg` and `fft1` is over 10%.

4.2 Scheduling Mix Analysis

Performance of multithreaded multicore processors depends on the mix of programs that are running at the same time. In other words, the time that a program takes to run will not depend only on its computation requirements but also on the ones of its co-runners. Therefore, the scheduler not only must select the task to be launched (e.g., a critical task) but also the appropriate core. The core must be selected according to the computational requirements of the tasks already running in each core.

For instance, suppose that the scheduler sends two tasks with a large execution time to core A and two shorter tasks to core B. Then, when both tasks in core B complete execution, the ones in core A will be still running. Therefore, the processor speed cannot be decreased, so wasting energy since core B is not performing any work. On the other hand, when tasks are fairly balanced among cores (so that long tasks are equally running in both cores), then the processor speed can be decreased as soon as the shorter tasks finish their execution.

This example, illustrates two opposite scheduling strategies. We refer as *fair balance* to the policy that launches complementary tasks to same core and as *poor balance* to the policy that does not take into account how complementary tasks are.

Table 5 shows the results for the same mix scheduled using a *fair balance* and a *poor balance* policies. Columns F400, F200, and F100 refer to the percentage of time that the processor runs at 400MHz, 200MHz and 100MHz respectively. The mix is composed by the *decode, encode, mpeg, rawd, adpcm* and *crc* benchmarks. In the fair balance policy, one core receives the *decode, mpeg, adpcm* benchmarks and the other one the *encode,rawd and crc* benchmarks. In this way, both the long ones as the short ones are equally distributed among cores. As opposite, in the *poor balance policy*, the long ones run at the same core. As a consequence, the processor is nearly all the time running at its maximum frequency (i.e., F400 $> 90\%$). As observed, the percentage of time that the system is working at its maximum speed is strongly related with the inter-arrival time length (i.e., the factor K explained in Section 3). Finally, notice that for a given mix, a higher percentage of time accessing to memory incurs a higher percentage of time of the system working at its maximum speed.

4.3 Estimating Energy Savings

As the time running at a given frequency depends on the scheduled mix, the energy savings will show

Table 5. Using different workload balancing strategies.

Scheduling policy	K	Execution time (M)	Memory time (%)	F400 (%)	F200 (%)	F100 (%)
Fair	1	59.10	0.71	0.94	0.05	0.01
	1.5	71.67	0.57	0.60	0.37	0.03
	2	83.03	0.49	0.40	0.58	0.02
	2.5	88.39	0.47	0.33	0.63	0.04
	3	90.93	0.45	0.30	0.66	0.04
Poor	1	56.44	0.69	0.97	0.01	0.02
	1.5	55.42	0.62	0.95	0.03	0.02
	2	61.21	0.57	0.93	0.03	0.04
	2.5	64.24	0.57	0.91	0.04	0.05
	3	67.45	0.40	0.90	0.05	0.05

Table 6. Mixes.

Mix Name	Order of the used Benchmarks
Mix 0	crc, ludcmp, adpcm, fir, fdct, fft1
Mix 1	fft1, crc, fir, ludcmp, adpcm, fdct
Mix 2	crc, fft1, fir, adpcm, ludcmp, fdct
Mix 3	fir, ludcmp, crc, fdct, fft1, adpcm
Mix 4	crc, adpcm, fdct, ludcmp, fir, fft1
Mix 5	ludcmp, adpcm, fdct, fft1, crc, fir
Mix 6	ludcmp, crc, fir, fdct, fft1, adpcm
Mix 7	fft1, fir, adpcm, ludcmp, crc, fdct

the same dependency. This means that the scheduling algorithm must be clever enough to discern between long and short tasks. To explore different scenarios about the energy gains that the algorithm could provide, different mixes have been designed. All of them fall in the *fair balance* policy since all submit at least one long benchmark to a given core. Table 6 shows the evaluated mixes.

Regarding frequency and voltage relationships, as mentioned in Section 3, we assume that the processor can work using three frequency speeds, each one using a different voltage level. Table 7 shows the assumptions about the energy consumed per cycle when working a 400MHz, 200MHz and 100MHz respectively. These values have been chosen according to the ones of the Pentium M processor [17, 30].

Power results were obtained running each benchmark until the simulator commits 3 milions of instructions. For each execution, we measured the number of cycles that the system works in a given frequency and multiplied this value by the corresponding energy per cycle.

For power comparisons purposes, for each factor, we have considered as unit the most consuming mix (i.e, m6) when running the processor at its maximum frequency, then, relative power savings are obtained for the remaining mixes when applying both the power-aware algorithm and the simple scheduler. Figure 2 plots the results.

Three main conclusions can be drawn:

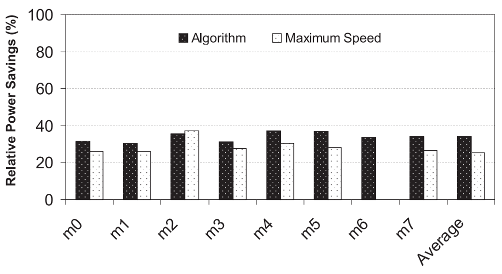
- Regarding the proposed algorithm, in general, power savings are higher as wider the inter-arrival time (i.e., long periods). This is because of the number of tasks in the system varies more frequently, thus, the algorithm is applied much more times. For instance, if the system would be overloaded most of the time, the algorithm would bring minor benefits.

- Independently of the inter-arrival time (factor 1, 1.5 and 2), the scheduling policy (either the algorithm is applied or not) allows to achieve power savings. Results show that only varying the scheduling policy, power savings could provide benefits, on the average, by about 25%, 61%, and 44%, for factor 1, factor 1.5, and factor 2, respectively.

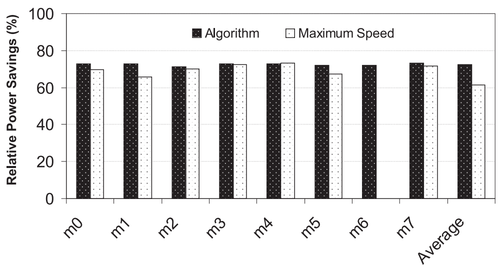
- The benefits of the algorithm not only depend on the algorithm itself but also on the baseline scheduler, that is, it might happen that applying the algorithm with a bad base task selection policy, could increase the power consumption. However, combining the algorithm with a given baseline scheduler (i.e., the same mix in the Figure) the algorithm always brings important power savings. Results show that, on average, these benefits are 34%, 73%, and 74%, for factor 1, factor 1.5, and factor 2, respectively.

Table 7. Energy spending by Frequency.

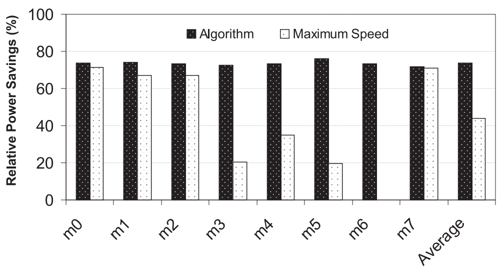
frequency [MHz]	400	200	100
energy [pJ/cycle]	349.2	186.3	123.8



(a) Factor 1



(b) Factor 1.5



(c) Factor 2

Figure 2. Relative Power.

5 Conclusions

This paper has introduced a novel power-aware scheduling algorithm for a state-of-the-art coarse-grain multithreaded multicore processor addressing soft real-time applications. The proposed algorithm applies dynamic voltage and frequency scaling, and adjust the processor speed to the running soft real-time workload. The scheduling algorithm has been evaluated using different mixes of benchmarks on a bi-core multithreaded embedded processor.

Experimental results show that power-aware scheduling algorithms can be designed for a given frequency and voltage level, however, if dynamic voltage scaling techniques are applied the power savings can be much larger. Power savings have been explored in three different scenarios (varying the inter-arrival time) and experimental results show that, on average, savings provided by the proposed algorithm are about 34%, 18%, and 67% across the scenarios.

As for future work we plan to design and evaluate schedulability tests for hard real-time tasks and their combination with soft real-time tasks.

Acknowledgments

This work was supported by Generalitat Valenciana under Grant GV06/326, Spanish CICYT under Grant TIN2006-15516-C04-01, CONSOLIDER-INGENIO 2010 under Grant CSD2006-00046 and Universidad Politécnica de Valencia under Grant PAID-06-07-20080029.

References

[1] T. AlEnawy and H. Aydin. Energy-aware task allocation for rate monotonic scheduling. In *Proceedings of the 11th IEEE Real Time on Embedded Technology and Applications Symposium*, pages 213–223, Washington, DC, USA, 2005. IEEE Computer Society.

[2] A. Anantaraman, K. Seth, K. Patil, E. Rotenberg, and F. Mueller. Virtual simple architecture (visa): Exceeding the complexity limit in safe real-time systems. In *Proceedings of the 30th International Symp. On Computer Architecture*, pages 350–361, 2003.

[3] J. Anderson and S. Baruah. Energy-efficient synthesis of periodic task systems upon identical multiprocessor platforms.

[4] H. Aydin and Q. Yang. Energy-aware partitioning for multiprocessor real-time systems. *Proceedings of the 17th International Parallel and Distributed Processing Symposium (IPDPS'03), Workshop on Parallel and Distributed Real-Time Systems*, 2003.

[5] F. J. Cazorla, P. M. Knijnenburg, R. Sakellariou, E. Fernández, A. Ramirez, and M. Valero. Predictable performance in smt processors: Synergy between the os and smts. *IEEE TRANSACTIONS ON COMPUTERS*, 55(7), 2006.

[6] E. M. B. Consortium. Automotive and telecomunication benchmark suites. Technical report, [Online]. Available: http://www.eembc.org/.

[7] I. Corp. Intel-xeon. Technical report, [Online]. Available: http://www.intel.com/products/processor/xeon.

[8] T. Corp. Transmeta crusoe. Technical report, [Online]. Available: http://www.transmeta.com.

[9] A. Corporation. Amd duron tm. Technical report, [Online]. Available: http://www.amd.com.

[10] J. Donald and M. Martonosi. Techniques for multicore thermal management: Classification and new exploration. In *ISCA '06: Proceedings of the 33rd annual international symposium on Computer Architecture*, pages 78–88, Washington, DC, USA, 2006. IEEE Computer Society.

[11] A. El-Haj-Mahmoud, A. S. AL-Zawawi, A. Anantaraman, and E. Rotenberg. Virtual multiprocessor: an analyzable, high-performance architecture for real-time computing. In *Proceedings of the 2005 International Conference on Compilers, Architectures and Synthesis for Embedded Systems*, pages 213–224, New York, NY, USA, 2005. ACM.

[12] G. Hinton and D. S. M. U. et al. The microarchitecture of the pentium 4 processor. intel technology journal. 2001.

[13] K. Hirata and J. Goodacre. Arm mpcore; the streamlined and scalable arm11 processor core. *Design Automation Conference, 2007. ASP-DAC '07. Asia and South Pacific*, pages 747–748, Jan. 2007.

[14] C.-M. Hung, J.-J. Chen, and T.-W. Kuo. Energy-efficient real-time task scheduling for a dvs system with a non-dvs processing element. In *Proceedings of the 27th IEEE International Real-Time Systems Symposium*, pages 303–312, 2006.

[15] R. Kalla, B. Sinharoy, and J. Tendler. Ibm power5 chip: a dual-core multithreaded processor. *Micro, IEEE*, 24(2):40–47, Mar-Apr 2004.

[16] P. Kongetira, K. Aingaran, and K. Olukotun. Niagara: a 32-way multithreaded sparc processor. *Micro, IEEE*, 25(2):21–29, March-April 2005.

[17] K. Krewell. Pentium m hits the street. microprocessor report. Technical report, [Online]. Available: http://www.transmeta.com, March 2003.

[18] C. Lee, M. Potkonjak, and W. Mangione-Smith. Mediabench: A tool for evaluating and synthesizing multimedia and communications systems. [Online]. Available: http://euler.slu.edu/ fritts/mediabench/mb1/.

[19] A. Limited. Arm11 mpcore processor. technical reference manual. Technical report, 2006.

[20] M. Marinoni and G. Buttazzo. Elastic dvs management in processors with discrete voltage/frequency modes. *IEEE Transactions on Industrial Informatics*, 3(1), 2007.

[21] C. McNairy and R. Bhatia. Montecito: a dual-core, dual-thread itanium processor. *Micro, IEEE*, 25(2):10–20, March-April 2005.

[22] A. Qadi, S. Goddard, and S. Farritor. A dynamic voltage scaling algorithm for sporadic tasks. *proceedings of the 24th IEEE Real-Time Systems Symp., Cancun, Mexico*, pages 52–62, 2003.

[23] V. Sharma, A. Thomas, T. F. Abdelzaher, K. Skadron, and Z. Lu. Power-aware qos management in web servers. In *Proceedings of the 24th IEEE Real-Time Systems Symposium, Cancun, Mexico*, pages 52–63, 2003.

[24] M. R. time Research Center. Wcet analysis project. wcet benchmark programs. Technical report, [Online]. Available: http:/www.mrtc.mdh.se/projects/wcet/benchmarks.html, 2006.

[25] D. Tullsen, S. Eggers, and H. Levy. Simultaneous multithreading: Maximizing on-chip parallelism. *Proceedings of the 22nd Annual International Symposium on Computer Architecture*, pages 392–403, 22-24 Jun 1995.

[26] R. Ubal, J. Sahuquillo, S. Petit, H. Hassan, and P. López. Power reduction in advanced embedded ipc processors. *International Journal of Intelligent Automation and Soft Computing*, Special Issue on Embedded System and Software for Intelligent Pervasive Computing(In Press).

[27] R. Ubal, J. Sahuquillo, S. Petit, and P. López. A simulation framework to evaluate multicore-multithreaded processors. In *Proceedings of the 19th International Symposium on Computer Architecture and High Performance Computing*, 2007.

[28] I. Ubicom. The ubicom ip3023 wireless network processor. Technical report, 2003.

[29] M. Verma, L. Wehmeyer, and P. Marwedel. Cache-aware scratchpad-allocation algorithms for energy-constrained embedded systems. *IEEE Transactions on CAD of Integrated Circuits and Systems*, 25(10):2035–2051, 2006.

[30] R. Watanabe, M. Kondo, M. Imai, H. Nakamura, and T. Nanya. Task scheduling under performance constraints for reducing the energy consumption of the gals multi-processor soc. In *Proceedings of the Design Automation and Test in Europe*, 2007.

[31] Y. Zhu and F. Mueller. Feedback edf scheduling of real-time tasks exploiting dynamic voltage scaling. *Real-Time Systems Journal*, 31(1-3):33–63, December 2005.