Trade-Offs Between Voltage Scaling and Processor Shutdown for Low-Energy Embedded Multiprocessors*

Pepijn de Langen and Ben Juurlink

Delft University of Technology, Computer Engineering Lab.
Mekelweg 4, 2628 CD Delft, The Netherlands
{pepijn,benj}@ce.et.tudelft.nl

Abstract. When peak performance is unnecessary, Dynamic Voltage Scaling (DVS) can be used to reduce the dynamic power consumption of embedded multiprocessors. In future technologies, however, static power consumption is expected to increase significantly. Then it will be more effective to limit the number of employed processors, and use a combination of DVS and processor shutdown. Scheduling heuristics are presented that determine the best trade-off between these three techniques: DVS, processor shutdown, and finding the optimal number of processors. Experimental results show that our approach reduces the total energy consumption by up to 25% for tight deadlines and by up to 57% for loose deadlines compared to DVS. We also compare the energy consumed by our scheduling algorithm to two lower bounds, and show that our best approach leaves little room for improvement.

1 Introduction

Recently, (single-chip) multiprocessors such as the IBM/Sony/Toshiba Cell architecture [1] and Philips Wasabi [2] have been introduced or announced for the high-performance embedded market. For such systems, the energy consumption is an important design consideration. The power consumption of a processor consists of a dynamic part (due to switching activity) and a static part (due to leakage current). In past technologies, the dynamic power has been much larger than the static power. With each technology generation, however, the leakage current is predicted to increase by a factor of five [3] and is predicted to surpass the dynamic power consumption [4].

In this paper we consider the problem of scheduling tasks on a multiprocessor system to minimize the total energy consumption. When the dynamic power dominates the static power, an effective technique to reduce the energy is to schedule the tasks on as many processors as possible to reduce the makespan of the schedule. Thereafter, the remaining time before the deadline (the *slack*) is used to scale down the supply voltages and operating frequencies. We refer to this technique as *Schedule-and-Stretch* (S&S).

When the static and dynamic power are comparable, however, S&S is no longer effective because it increases the amount of leakage current by using more processors than necessary and by lengthening the time it takes to complete the computation. In previous work [5] we have proposed LAMPS (Leakage-Aware MultiProcessor Scheduling).

* This research was supported in part by the Netherlands Organisation for Scientific Research (NWO).

S. Vassiliadis et al. (Eds.): SAMOS 2007, LNCS 4599, pp. 75–85, 2007.

LAMPS does not employ as many processors as possible to maximize the amount of slack, but determines an optimal balance between the number of processors that should be used and the level of frequency/voltage scaling.

In this work, we extend both S&S and LAMPS in the following ways. First, we assume discrete voltage levels, while in [5] we have assumed that any voltage/frequency level can be used. Second, we extend both heuristics with the option to shut down processors temporarily. Third, we include two lower bounds, one for the case where only a single frequency is used for all tasks, and one for the case where processors can run at different frequencies and these frequencies may change over time.

Experimental results show that our best approach reduces the total energy consumption by up to 25% for tight deadlines (1.5x the critical path length) and by up to 57% for loose deadlines (8x the critical path length) compared to S&S and by up to 14% respectively 11% compared to LAMPS. Comparing these results to the theoretical bounds indicates there is little room left for improvement. More specifically, for fairly coarse-grain task graphs LAMPS+PS attains over 94% of the possible savings, provided the frequency is the same for all active processors and is constant throughout the schedule.

This paper is organized as follows. Section 2 contains an overview of related work. The power model employed in this work, dynamic voltage scaling, and processor shutdown are explained in Section 3. Section 4 reviews S&S and LAMPS and presents our novel scheduling heuristics that extend S&S and LAMPS with the possibility to shut down processors for a period of time. Experimental results are provided in Section 5. In Section 6, conclusions are drawn and some directions for future research are given.

2 Related Work

Applying DVS to multiprocessor scheduling has been investigated by a significant number of researchers. An overview is provided by Jha [6]. As described in Section 1, one approach is to use an existing scheduling algorithm, such as list scheduling with earliest deadline first (EDF), to finish the tasks as early as possible and use the remaining slack before the deadline to lower the supply voltage. This technique has been proposed by several authors [7,8] using different names and, therefore, we refer to it as *Schedule and Stretch* (S&S). Leakage current was not included in their energy calculations, however.

Jejurikar et al. [9] showed that there is an optimal operating point, called the critical speed, at which the total energy consumption is minimized. Lowering the supply voltage below this point increases the energy consumption. They combined this knowledge with processor shutdown and DVS and used it for real-time scheduling. A similar approach was followed by [10], who employed a fixed priority instead of EDF. However, these works assumed that the tasks are independent and focussed on single-processor scheduling. The same model is assumed in [11] and [12]. In addition, the first did not consider DVS, and the second assumed a continuous voltage range.

In other work [13], the scheduling is done in a way to optimize the possibilities for selecting different voltages. Varatkar et al. [14] tried to execute part of the code on a lower supply voltage while minimizing communication. Some researchers have proposed to improve DVS by also adjusting the threshold voltage when scaling the

supply voltage [15,16,17,18]. None of these works, however, attempted to determine the number of processors that yields the least energy consumption.

Xu et al. [19] proposed to minimize energy consumption by both using a combination of DVS and choosing the correct number of employed processors. Their work, however, targets embedded clusters in which the nodes provide the same type of service in a client-server model. Furthermore, these authors do not consider static scheduling but instead propose an online algorithm similar to [8].

Our work differs in the following ways. First, we focus on static multiprocessor scheduling, where others mainly focussed on single-processor or real-time multiprocessor scheduling. Second, we use a detailed power model and limit the voltage to discrete steps. Third, we consider DVS as well as processor shutdown and finding the correct number of processors. Fourth, we use a publicly available set of task graphs, where most others have used randomly generated graphs.

3 Preliminaries

In this section we describe the employed models, as well as two primary ways to reduce power dissipation: dynamic voltage scaling and processor shutdown.

System and Application Model. We assume a shared memory multiprocessor system running parallel applications, for which the scheduling and mapping are statically determined. The applications are represented as weighted directed acyclic graphs (DAGs), where nodes correspond to tasks, edges to task dependences, and node weights to task processing times. We furthermore assume that this system is CPU bound. As explained by Liberato et al. [20], real-time applications with periodic tasks can be translated to DAGs using the *frame-based scheduling* paradigm.

Another common application model is Kahn Process Networks [21], where a group of processes are connected by communication channels to form a network of processes. Each process is in principle infinite and receives data over its input channels, processes it, and sends the results over the output channels. Here there is not a single deadline but a certain throughput must be guaranteed. This model can be converted to DAGs by making several copies of the KPN, by adding an edge from each node in the ith copy to the corresponding node in the $(i+1)$st copy, and by assigning the output nodes of the first copy an arbitrary but reasonable deadline. The deadline of the output nodes of each successive copy is set to the deadline of the corresponding node in the previous copy plus the reciprocal of the throughput. A simple example is depicted in Fig. 1.

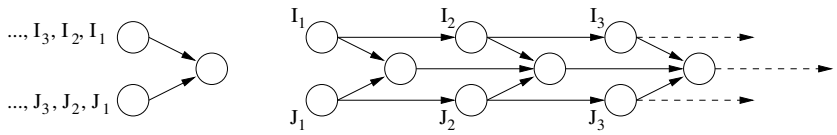


Fig. 1. Simple example for translating KPNs into DAGs

Mainly due to unpredictable behavior in the memory system, the execution time of a task does not solely depend on the clock frequency. However, since reducing the frequency will make memory accesses relatively less costly, it is safe to assume that executing a task on $1/N^{th}$ of the frequency will take at most N times as much time.

Power Model. In this work, we use the power model described in [9], which in turn is based on the model and parameters given in [16], where it has been verified with SPICE simulations. In this model, the power consumption of a processor is given by:

$$P = P_{\text{AC}} + P_{\text{DC}} + P_{\text{on}},$$

where P_{AC} is the dynamic power consumption (due to switching activity), P_{DC} is the static power consumption (due to leakage current), and P_{on} is the intrinsic power consumption needed to keep the processor on. Like [9], we assume P_{on} is $0.1W$. The dynamic and static power are given by:

$$P_{\text{AC}} = C_{\text{eff}} V_{\text{dd}}^2 f \qquad \text{and} \qquad P_{\text{DC}} = V_{\text{dd}} I_{\text{subn}} + |V_{\text{bs}}| I_j,$$

where C_{eff} is the effective switching capacitance, V_{dd} is the supply voltage, f is the operating frequency, I_{subn} is the sub-threshold leakage current, V_{bs} is the voltage applied between body and source, and I_j is the reverse bias junction current. The sub-threshold leakage current and the threshold voltage are given by:

$$I_{\text{subn}} = K_3 e^{K_4 V_{\text{dd}}} e^{K_5 V_{\text{bs}}} \qquad \text{and} \qquad V_{\text{th}} = V_{\text{th1}} - K_1 V_{\text{dd}} - K_2 V_{\text{bs}},$$

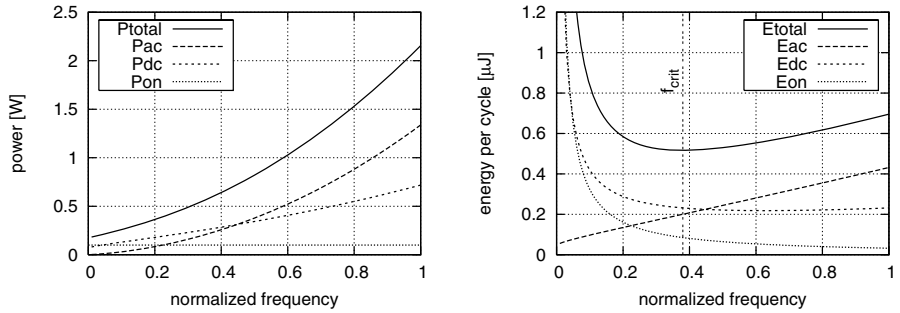
where $K_1 \ldots K_5$ and V_{th1} are constants. Finally, the relation between operating frequency, supply voltage, and threshold voltage is:

$$f = (V_{\text{dd}} - V_{\text{th}})^{\alpha} / L_d K_6,$$

where L_d represents the logic depth and K_6 and α are constants for a certain technology. We use the same constants for $70nm$ technology as [9,16], which are omitted due to space constraints. The maximum frequency of this processor is 3.1GHz, which requires a supply voltage of 1V. Figures 2(a) and 2(b) depict the resulting power consumption and energy per cycle as a function of the normalized operating frequency.

Dynamic Voltage Scaling. DVS mainly reduces the dynamic power consumption, which increases quadratically with the supply voltage. The static component, although having a exponential relation with supply voltage, does not decrease as much with decreasing supply voltage as the dynamic component, as is depicted in Fig. 2(a).

Since energy equals power times time, the energy consumption will actually start to increase if the frequency is decreased below a certain point. Fig. 2(b) depicts the energy per cycle as a function of the normalized frequency. It can be seen that the *optimal* or *critical* frequency (f_{crit}) is 0.38 times the maximum. Because of the discrete voltage levels, however, the critical frequency is reached at a supply voltage of 0.7V, corresponding to a normalized frequency of 0.41. Scaling below this frequency will reduce the power consumption but not the total energy consumption, provided that the processors can be shut down for the remaining time. When there is no sleep/shutdown mode, scaling below f_{crit} will, in fact, reduce the total energy consumption, since the processors also consume energy for the remaining time.



(a) Power consumption. (b) Energy consumption.

Fig. 2. Power and energy consumption as a function of the normalized frequency

Processor Shutdown. The second technique to reduce the energy consumption of a multiprocessor system is to put idle processors temporarily in a deep sleep or shutdown mode. The advantage of this technique over DVS is that it reduces all terms of the total power consumption, not only the dynamic part. When shutting down a processor, however, the contents of, e.g., caches and branch predictors are lost. When a processor is switched back on, they have to be warmed up again, which causes additional delay and consumes extra energy. We use the estimates of Jejurikar et al. [9], who estimated that a processor in sleep state consumes about $50\mu W$ of power and that shutting down and resuming a processor incurs an energy overhead of $483\mu J$. This overhead includes the supply voltage switching as well as the energy spent to warm up caches and predictors. The additional delay incurred by powering down can be hidden by waking up the processor a short time before the end of the idle period.

Processor shutdown is only beneficial if a processor is idle for a sufficiently long period. Since in most cases applications with rather fine-grain tasks will have relatively short idle periods (unless the task graph is very unbalanced), such applications will in general not benefit from to shutting down processors temporarily between the execution of two tasks. However, it might still be energy efficient to shut down at the end of the schedule, provided the deadline is relatively long.

4 Scheduling for Energy Minimization

In this section we review S&S and LAMPS and enhance both scheduling approaches with the option to shut down processors temporarily. In the schedules produced by these approaches, all processors run at the same operating frequency and this frequency is constant throughout the whole schedule. Both S&S and LAMPS employ list scheduling with earliest deadline first (LS-EDF) to perform the actual scheduling. EDF does not necessarily produce the best schedule, however. To investigate if other scheduling algorithms could result in additional energy gains, we also present an ideal model in which idle processors are assumed to consume no energy. Furthermore, we also show the improvements that could be attained if the frequency could vary among processors and over time.

S&S and S&S+PS. As described before, in S&S the task graph is first scheduled using LS-EDF to maximize the amount of slack before the deadline. Thereafter, this slack is exploited to lower the supply voltage. By employing this technique, the energy consumption can already be reduced by 30% for tight deadlines by more than 70% for loose ones [5].

We extend S&S with the option to shut down processors temporarily. We refer to this heuristic as *S&S with Processor Shutdown* (S&S+PS). In S&S+PS, the task graph is again first scheduled using the EDF policy. Thereafter, the optimal balance between processor slowdown (through DVS) and shutdown is determined by gradually scaling the operating frequency from the maximum frequency to the minimum frequency required to meet the deadline using discrete voltage level steps of 0.05V. For each frequency, the remaining slack both inside as well as at the end of the schedule is used to shut down processors, provided the idle period is greater than the minimum idle period to result in energy savings. In other words, the slack is only used to shut down a processor if it is large enough to make up for the additional energy consumption due to loss of state.

LAMPS and LAMPS+PS. In LAMPS, a trade-off is made between the number of processors that should be employed and the amount of voltage scaling. The remaining processors are turned off. The number of processors that minimizes the energy is found by calculating the energy consumption of the schedule produced by S&S for $N_{\min}$, $N_{\min}+1$, …, $N_{\max}$ processors, where $N_{\min}$ is the number of processors needed to finish before the deadline and $N_{\max}$ is the number of processors that can be used to reduce the makespan of the schedule.

We also enhance the LAMPS heuristic with the option to shut down processors and refer to the resulting heuristic as LAMPS+PS. As in LAMPS, the number of processors that minimizes the total energy consumption is determined by calculating the energy consumption for $N_{\min}, N_{\min}+1, \ldots, N_{\max}$ processors, where $N_{\min}$ is the minimal number of processors needed to meet the deadline and $N_{\max}$ is the number of processors that can be employed efficiently. For each number of processors, we then determine the balance between DVS and processor shutdown by scaling the frequency from the maximum to the minimum frequency required to meet the deadline. For each frequency, we then use the available slack to shut down processors, similar to the S&S+PS heuristic.

LIMIT-SF & LIMIT-MF. In the approaches described above, the schedule is always produced by EDF. It is known, however, that EDF is suboptimal for multiprocessor scheduling. Furthermore, in the approaches the frequency is always constant throughout the entire schedule. To investigate if additional energy can be saved by employing a different scheduling algorithm or by allowing different frequencies, we also include two lower bounds, one for the case with a single frequency (LIMIT-SF) and one for the case where multiple frequencies are allowed (LIMIT-MF).

LIMIT-SF has the following characteristics. First, idle processors are assumed to consume no energy at all. In other words, only active cycles are considered when calculating the energy consumption and, consequently, there is no benefit from or penalty for shutting down processors. Second, the number of processors is equal to the number of tasks. Since idle processors consume no energy, using fewer processors will not

reduce the energy. Third, the frequency is scaled down to the optimal frequency if possible to meet the deadline, or otherwise as much as possible. No schedule can consume less energy than this ideal model, provided that the frequency is the same for all active processors and is constant throughout the schedule.

The difference between LIMIT-MF and LIMIT-SF is that in LIMIT-MF all tasks are scheduled at the critical frequency. Because of this and since idle processors are assumed to consume no energy, LIMIT-MF is an absolute lower bound, even for the case where processors can run at different speeds and where the frequency may change over time. We note, however, that it may happen that the schedule produced by LIMIT-MF does not meet the deadline.

Since both LIMIT-SF and LIMIT-MF do not depend on any particular scheduling algorithm, this implies that these results cannot be improved by employing a different scheduling algorithm than EDF.

5 Experimental Evaluation

In this section, we present and compare the results of the different scheduling approaches discussed in Section 4. We use the same power model as used by [16] and [9], as explained in Section 3. We again emphasize that a processor in sleep state consumes 50μW and that switching a processor off and on requires 483μJ of energy.

Experimental Setup. Table 1 lists the benchmarks that have been used, as well as the number of nodes and edges, the critical path length, and the sum of all node weights (total work). These benchmarks have been taken from the *Standard Task Graph Set* [22]. The first three have been derived from real applications, while the other three have been randomly generated. We note that most previous works have used only randomly generated task graphs to validate their approaches.

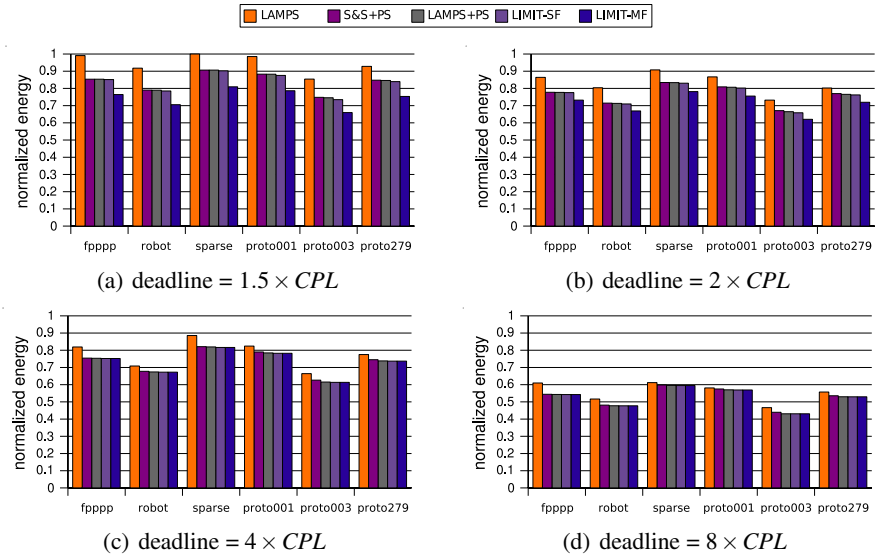
Table 1. Employed benchmarks and their main characteristics

name	number of nodes	number of edges	critical path	total work
fpppp	334	1196	1062	7113
robot	88	130	545	2459
sparse	96	128	122	1920
proto001	273	1688	167	4711
proto003	164	646	556	1599
proto279	1342	16762	735	13302

Since the Standard Task Graph Set does not provide deadlines, we have used deadlines of 1.5, 2, 4, and 8 times the critical path length (CPL) when running at the maximum frequency of 3.1GHz. It also does not define the unit of the task weights. Instead, the weights are given as integers in the range from 1 to 300. Therefore, two different scenarios are considered. In the first scenario, corresponding to rather coarse-grain tasks, a weight of 1 in a task graph implies an execution time of $3.1 \cdot 10^6$ cycles, which

is 1 millisecond when running at the maximum frequency of 3.1GHz. In the second scenario, corresponding to relatively fine-grain tasks, the same weight implies an execution time of $3.1 \cdot 10^4$ cycles, which at maximum frequency takes 10 microseconds.

Experimental Results. Figs. 3 and 4 depict the results for coarse grain and fine grain tasks, respectively. For each scenario, we show the energy consumption for deadlines of 1.5, 2, 4, and 8 times the CPL. Each figure shows the results of the four different approaches explained in Section 4, as well as the theoretical limits. Throughout this section, S&S is used as the baseline against which we compare the other heuristics.



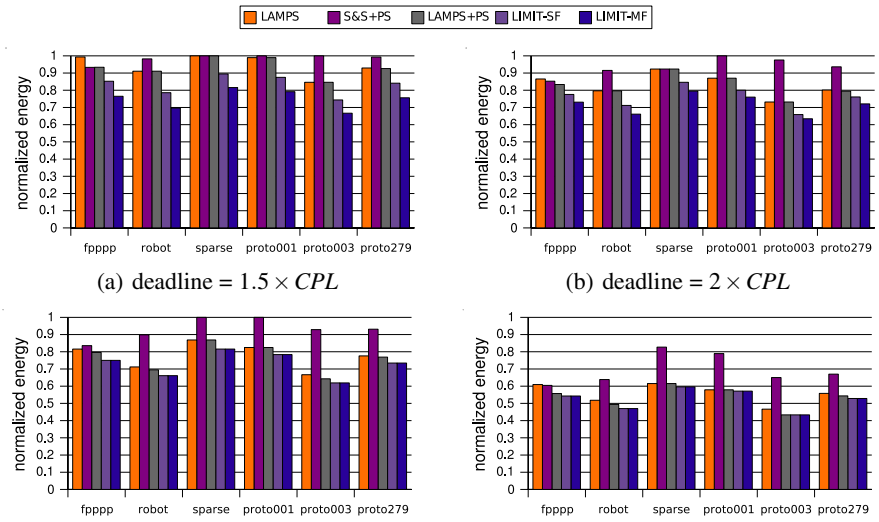
(a) deadline = $1.5 \times CPL$ (b) deadline = $2 \times CPL$

(c) deadline = $4 \times CPL$ (d) deadline = $8 \times CPL$

Fig. 3. Energy consumption for coarse-grain tasks

First we compare the energy consumption of the schedules produced by LAMPS to the energy consumption of the schedules generated by S&S. Figs. 3 and 4 show that LAMPS improves upon S&S mainly for less strict deadlines. This can be expected because for tight deadlines (1.5x the CPL), LAMPS requires the same or nearly the same number of processors as S&S to meet the deadline, and therefore consumes the same or nearly the same amount of energy as S&S. In other words, if the deadline is tight, there is less opportunity to turn off processors. For loose deadlines (8x the CPL), on the other hand, LAMPS consumes significantly less energy than S&S, simply because it can employ fewer processors. In this case LAMPS reduces the total energy consumption by 45% on average compared to S&S with a maximum of 53%. For fine-grain tasks, depicted in Fig. 4, the relative differences between S&S and LAMPS are the same as with coarse-grain tasks, since both heuristics do not shut down processors. Compared to [5] the energy savings are generally slightly smaller because there a continuous voltage range was assumed while in this work discrete voltage levels are assumed.

We now compare S&S+PS to S&S. Because S&S employs a large number of processors, it consumes a significant amount of static power. Therefore, S&S+PS improves



(a) deadline = $1.5 \times CPL$ (b) deadline = $2 \times CPL$

(c) deadline = $4 \times CPL$ (d) deadline = $8 \times CPL$

Fig. 4. Energy consumption for fine-grain tasks

upon S&S significantly, by shutting down idle processors temporarily. The gains, in this case, are considerably larger for coarse-grain tasks (30% on average) than for fine-grain tasks (12% on average), because in the latter case the slack is often not large enough to make shutdown beneficial.

LAMPS+PS improves upon LAMPS mostly for coarse-grain tasks. Again, the main reason for this is that for fine-grain tasks, the slack is often not large enough to make shutting down worthwhile. With coarse grain tasks, however, a significant amount of energy can be saved by shutting processors down temporarily. The improvement of LAMPS+PS over LAMPS is typically less than the improvement of S&S+PS over S&S. This is because in LAMPS the static dissipation is already reduced by using a smaller number of processors compared to S&S. For coarse-grain tasks, the maximum improvements by LAMPS+PS upon LAMPS are 14% and 11%, for deadlines of $1.5\times$ and $8\times$ the CPL respectively.

For coarse-grain tasks, the total improvement by LAMPS+PS upon S&S is 29% on average, with a maximum of 25% for deadlines of $1.5\times$ the CPL and a maximum of 57% for deadlines of $8\times$ the CPL. For fine-grain tasks, LAMPS+PS improves upon S&S by 25% on average, with a maximum of 15% for deadlines of $1.5\times$ the CPL and a maximum of 57% for deadlines of $8\times$ the CPL.

LIMIT-SF in Figs. 3 and 4 gives an upper limit on the energy savings using our current single-frequency model. Using S&S as the baseline and LIMIT-SF as the maximum, it shows that LAMPS+PS attains more than 94% of the possible energy reduction with coarse-grain tasks, for all combinations of benchmarks and deadlines. For fine-grain tasks and strict deadlines ($1.5\times$ the CPL), LAMPS+PS achieves more than 41% of the potential savings on 4 out of the 6 benchmarks. With less strict deadlines, LAMPS+PS attains more than 50% of the possible savings on all benchmarks.

In Figs. 3 and 4, Limit-MF is an indication for the possible improvements that could be attained by allowing the processors to run at a different frequency, and by allowing these frequencies to change over time. The results indicate that there is very little room for improvements when the deadline is relatively loose. For stricter deadlines, some savings may be attained, but mostly for fine-grain tasks. In the case of fine-grain tasks with strict deadlines, the periods of inactivity are often too small to make shutting down worthwhile. In this case, allowing varying frequencies might result in some additional savings. However, when the deadline is less strict and/or the task graph is fairly coarse-grained, shutting down processors becomes worthwhile. In this case, scheduling tasks at different frequencies will not provide a significant improvement.

6 Conclusions and Future Work

As feature sizes keep decreasing, the contribution of leakage current to the total energy consumption is expected to increase. Depending on the amount of slack that remains before the deadline, the amount of parallelism, and the granularity of the application, voltage scaling as well as shutting down processors can be used to reduce the energy significantly. At the same time, it is important not to employ to many processors.

We have shown that LAMPS+PS reduces the total energy by up to 25% for tight deadlines and up to 57% for loose ones compared to the S&S algorithm. For coarse-grain tasks and a single frequency, LAMPS+PS attains more than 94% of the possible energy reduction, i.e., the energy reduction achieved by LIMIT-SF compared to S&S. Since LIMIT-SF is independent of the scheduling algorithm, this implies that there is almost no room left for improvement by using other scheduling algorithms than EDF.

Even when multiple frequencies are allowed, LAMPS+PS reduces the energy consumption close to the theoretical limit (LIMIT-MF). For loose deadlines ($4\times$ or $8\times$ the critical path length), LIMIT-MF consumes the same amount of energy as LIMIT-SF, and so LAMPS+PS again attains over 94% of the potential savings with coarse-grain tasks. Averaged over all tested deadlines, our best approach still attains over 84% of the possible savings. As a result, it will be nearly impossible to reduce the overall energy consumption further by using other scheduling algorithms that produce schedules in which different processors can run at different frequencies and in which the frequency can change over time. Applications consisting of relatively fine-grain tasks, on the other hand, might benefit from using other scheduling approaches. However, since LIMIT-MF does not take the deadline into account, real scheduling approaches will probably not reach this limit. Consequently, the actual benefit from having multiple frequencies will probably be much less. We intend to investigate the impact of multiple frequencies and other scheduling algorithms in future research.

References

1. Hofstee, H.: Power Efficient Processor Architecture and the Cell Processor. In: Proc. Int. Symp. on High-Performance Computer Architecture, pp. 258–262 (2005)
2. Stravers, P., Hoogerbrugge, J.: Homogeneous Multiprocessing and the Future of Silicon Design Paradigms. In: Proc. Int. Symp. on VLSI Technology, Systems, and Applications, pp. 184–187 (2001)

3. Borkar, S.: Design Challenges of Technology Scaling. IEEE Micro 19(4), 23–29 (1999)
4. Duarte, D., Vijaykrishnan, N., Irwin, M., Tsai, Y.: Impact of Technology Scaling and Packaging on Dynamic Voltage Scaling Techniques. In: Proc. IEEE Int. ASIC/SOC Conf, IEEE Computer Society Press, Los Alamitos (2002)
5. de Langen, P., Juurlink, B.: Leakage-Aware Multiprocessor Scheduling for Low Power. In: Proc. Int. Parallel and Distributed Processing Symp. (2006)
6. Jha, N.: Low-Power System Scheduling, Synthesis and Displays. IEE Proc. on Computers and Digital Techniques 152(3), 344–352 (2005)
7. Gruian, F., Kuchcinski, K.: LEneS: Task Scheduling for Low-Energy Systems Using Variable Supply Voltage Processors. In: Proc. Conf. on Asia South Pacific Design Automation, pp. 449–455 (2001)
8. Zhu, D., Melhem, R., Childers, B.: Scheduling with Dynamic Voltage/Speed Adjustment Using Slack Reclamation in Multiprocessor Real-Time Systems. IEEE Trans. on Parallel and Distributed Systems 14(7), 686–700 (2003)
9. Jejurikar, R., Pereira, C., Gupta, R.: Leakage Aware Dynamic Voltage Scaling for Real-Time Embedded Systems. In: Proc. Conf. on Design Automation, pp. 275–280 (2004)
10. Quan, G., Niu, L., Hu, X.S., Mochocki, B.: Fixed Priority Scheduling for Reducing Overall Energy on Variable Voltage Processors. In: Proc. Int. Real-Time System Symposium, pp. 309–318 (2004)
11. Lee, Y., Reddy, K., Krishna, C.: Scheduling Techniques for Reducing Leakage Power in Hard Real-Time Systems. In: Proc. Euromicro Conf. on Real-Time Systems, pp. 105–112 (2003)
12. Irani, S., Shukla, S., Gupta, R.: Algorithms for Power Savings. In: ACM-SIAM Symp. on Discrete Algorithms, pp. 37–46 (2003)
13. Zhang, Y., Hu, X.S., Chen, D.Z.: Task Scheduling and Voltage Selection for Energy Minimization. In: Proc. Conf. on Design Automation, pp. 183–188 (2002)
14. Varatkar, G., Marculescu, R.: Communication-Aware Task Scheduling and Voltage Selection for Total Systems Energy Minimization. In: Proc. Int. Conf. on Computer-Aided Design, pp. 510–517 (2003)
15. Gonzalez, R., Gordon, B., Horowitz, M.: Supply and Threshold Voltage Scaling for Low Power CMOS. IEEE Journal of Solid-State Circuits 32(8), 1210–1216 (1997)
16. Martin, S., Flautner, K., Mudge, T., Blaauw, D.: Combined Dynamic Voltage Scaling and Adaptive Body Biasing for Lower Power Microprocessors under Dynamic Workloads. In: Proc. Int. Conf. on Computer-Aided Design, pp. 721–725 (2002)
17. Andrei, A., Schmitz, M., Eles, P., Peng, Z., Al-Hashimi, B.M.: Overhead-Conscious Voltage Selection for Dynamic and Leakage Energy Reduction of Time-Constrained Systems. In: Proc. Conf. on Design, Automation and Test in Europe, pp. 518–525 (2004)
18. Yan, L., Luo, J., Jha, N.K.: Combined Dynamic Voltage Scaling and Adaptive Body Biasing for Heterogeneous Distributed Real-time Embedded Systems. In: Proc. Int. Conf. on Computer-Aided Design, pp. 30–37 (2003)
19. Xu, R., Zhu, D., Rusu, C., Melhem, R., Moss, D.: Energy-Efficient Policies for Embedded Clusters. In: Proc. ACM SIGPLAN/SIGBED Conf. on Languages, Compilers, and Tools for Embedded Systems. pp. 1–10 (2005)
20. Liberato, F., Lauzac, S., Melhem, R., Moss, D.: Fault Tolerant Real-Time Global Scheduling on Multiprocessors. In: Proc. Euromicro Conf. on Real-Time Systems. (1999) 252–259
21. Kahn, G.: The Semantics of a Simple Language for Parallel Programming. In: Information Processing, pp. 471–475 (1974)
22. Kasahara, H., Tobita, T., Matsuzawa, T., Sakaida, S.: Standard Task Graph Set, `http://www.kasahara.elec.waseda.ac.jp/schedule/`