Power-Aware Scheduling for Hard Real-Time Embedded Systems Using Voltage-Scaling Enabled Architectures

Amjad Mohsen and Richard Hofmann

Department of Computer Science 7, University of Erlangen-Nürnberg
91058 Erlangen, Germany
amjad.muhsen@informatik.uni-erlangen.de, rhofmann@cs.fau.de

Abstract. In this paper, we present a power-aware scheduling scheme for hard real-time embedded systems design. Our new approach can enhance the efficiency of both dynamic voltage scaling (DVS) and dynamic V_{th} scaling (DVTS). While optimizing the schedule in the time domain, the priorities of the tasks are modified dynamically based on their contribution to the overall power/energy reduction. The scheduling scheme leads to better "distribution" and "utilization" of slack intervals in the system which in return improves the efficiency of voltage scaling techniques. The voltage schedule is generated based on a global view of the components' energy profile when executing different tasks. The experimental results prove the applicability of our approach.

1 Introduction

Reducing power/energy consumption is a major design interest for mobile as well as for stationary digital systems. The constraint on the consumed power/energy override now other design constraints such as performance, especially in mobile environments. Hard real-time systems in mobile communication as well as in other application areas bring this topic to a real challenge: Delay constraints must be satisfied to guarantee correct operation while the consumed power/energy must be kept below a certain level for safety and reliability reasons.

To address this challenge, the design is optimized at different abstraction levels. Low-level approaches and tools can not deal with future chip complexity. Based on the fact that at high abstraction levels design tradeoffs are better understood, design decisions made at these levels cause drastic reduction in the consumed power and can also shorten the design cycle. Algorithm optimization, components' selection as well as application partitioning are critical issues in low power/energy design.

A widely used technique which can reduce the consumed power/energy is dynamic voltage scaling (DVS). It trades performance for power during system operation when peak performance is not essentially required [1], [2]: The supply voltage and the operational frequency are adapted under the control of the operating system based on the required performance. For applications with a pre-defined performance limits, we suggest to plan the required voltage levels to execute different tasks during system synthesis. As a result, the overhead of calculating or predicting the required voltage level(s) at run-time is reduced to a minimum.

Basically, more energy reduction can be achieved when longer slack intervals are utilized by DVS. Additionally, the way in which the available slack intervals are exploited directly influences the efficiency of DVS. The higher the energy profile the

V. Paliouras, J. Vounckx, and D. Verkest (Eds.): PATMOS 2005, LNCS 3728, pp. 127–136, 2005.

more energy reduction a task can achieve when applying DVS. Therefore, adapting the time schedule of the tasks such that tasks which are major energy consumers are enabled to utilize longer slack intervals can remarkably maximizes the saved energy when applying DVS.

In future technologies (*70 nm* and less), the leakage power becomes more dominant and will account for about 50% of the total consumed power or even more. Dynamic V_{th} scaling (DVTH) is suggested to reduce the leakage power by adaptively changing the body bias voltage [3]. The availability and the distribution of slack intervals are key issue also for this technique. Additionally, the amount of saved leakage power depends on the time duration the system spends in each operation mode (active, inactive). Modifying the time schedule of the tasks to take these issues into consideration can be a source of extra power reduction.

This paper is organized as follows: The next section presents a summary of selected related work. Section 3 overviews our low power/energy system-level co-synthesis approach. In section 4, the power-aware scheduling scheme is explained. Experimental results are presented and analyzed in section 5. Section 6 concludes this paper.

2 Related Work

Dynamic voltage scaling is used to trade performance for power without impacting the peak performance of the system. In [4], algorithms were presented to determine the needed operating speed of a processor at run-time under software control. The operating voltage was adjusted based on the expected needed performance.

A task scheduling heuristic based on list-scheduling was introduced in [5]. The objective was reducing the consumed energy in systems by applying DVS. The scheme dynamically calculated the tasks' priorities and chose the supply voltage levels that could minimize the consumed energy. The presented priority function was aware of energy but might fail under tight deadline. A priority function tuning mechanism was used to compensate for this drawback

A hybrid global/local search optimization approach in a multi-processor system for dynamic voltage scaling was presented in [6]. Although this approach could yield the voltage levels that could minimize energy, the optimization itself was time consuming. Also, the authors assumed that the schedule was computed and the order of tasks' execution was known in advance. Hence, the influence of scheduling on power reduction was not tackled.

A static voltage scheduling problem was proposed and formulated as an integer linear programming (ILP) problem in [7]. It was assumed that the tasks have different average switching capacitance per cycle to consider energy profiles of processing elements. The study showed that considering energy profiles when scaling the voltage is a source of extra power reductions. Other studies also reached a similar conclusion [8]. However, these studies were done for single processor systems.

Grajcar suggested a genetic list scheduling algorithm without tackling the power problem [9]. Schmitz et al. used this scheduling approach to achieve energy–efficient scheduling when using DVS-enabled architectures [10]. The authors suggested the genetic list-based scheduling algorithm inside a mapping optimization loop. The latter optimization loop was based on genetic algorithms.

A low power co-synthesis tool (LOPOCOS) was presented in [11]. The objective was to help the designer in identifying energy-efficient application partitioning for distributed embedded systems. This approach assumed heterogeneous and DVS-enabled architectures. Although it performs better than previously suggested approaches, for applications with stringent delay constraints, the amount of saved power/energy could be moderate.

Our approach for low power/energy co-design is comprehensive and targets real-time systems. Starting at the FDT level (formal description techniques), the design space is explored to find low power/energy implementation alternatives that satisfy the constraints. The core issue which is handled in this paper is the scheduling problem. When applying voltage scaling mechanisms, the scheduling problem becomes even more involved, because the voltage scaling mechanism performs well only if the time schedule is optimized for this purpose.

Due to this correlation, the time schedule is iteratively adapted in order to enhance the efficiency of voltage scaling mechanisms: Tasks' priorities are calculated dynamically based on their contribution to power/energy reduction. Tasks that cause higher power/energy reductions are given higher priority to execute in an attempt to increase the available slack intervals for these tasks. DVS and DVTS can benefit from this approach. Available slack intervals are efficiently exploited by optimizing the voltage schedule based on a global view of energy profiles of all tasks and their mappings.

3 System Co-synthesis and Voltage Scaling Aspects

System-level synthesis can be seen as mapping a behavioural description onto a structural specification. Functional objects have the granularity of algorithms, tasks, procedures, etc, while structural objects are processors, ASICs, buses, etc. This section briefly presents our system-level low power/energy co-synthesis approach. Then, some basic issues in voltage scaling are introduced.

3.1 System Co-synthesis

In our approach, system-level synthesis is considered as a multiobjective optimization problem: Performance, power, and cost are considered while optimizing allocation, binding, and scheduling. Further refinement steps are proposed in our approach to optimize the design under stringent performance constraints. The optimization process searches the design space to find implementations that satisfy the design constraints and performs a rating on them with respect to the optimization goal(s). Evolutionary-based design space exploration is adopted in our approach. For this purpose, we have integrated to our automated co-synthesis tool the well-known evolutionary multi-objective optimizer SPEA2 [12]. SPEA2 is responsible for assigning fitness values to individuals. Each individual encodes a possible implementation alternative. Fittest individuals are selected to the mating process.

System synthesis is performed automatically using an internal system model which consists of two graphs: A task graph (TG), and an architecture graph (AG). Both are automatically generated from the specification. The TG is a directed acyclic graph $F_p(\Psi,\Omega)$, where Ψ represents the set of vertices in the graph ($\psi_i \in \Psi$) and Ω is the set

of directed edges representing the precedence constraints and data dependencies ($\omega_i \in \Omega$). The AG is $F_A(\Theta, \Re)$, where Θ represents the available architectures ($\theta_i \in \Theta$) and ($\rho_i \in \Re$) represents the available connections between the hardware components. For each hardware component ($\theta_i \in \Theta$), a finite set of resource types (S) is defined. For each resource type ($s_i \in S$) there is a set of associated ratios (R_s) that specify the scaling of power, delay, and cost when using this type for the selected component. Hard real-time constraint(s) T_i are forced on a node or on a set of nodes as well as an absolute time constraint T_T.

3.2 Applying Voltage Scaling

DVS and DVTS are efficient techniques which can reduce the overall consumed power/energy. But some factors such as the availability of enough slack intervals, their distribution among tasks, and the way in which these intervals are exploited have crucial influence on the efficiency of these techniques.

In systems which employ DVS technology, the supply voltage and the operational frequency can be adapted based on the required performance when executing a task. Reducing the supply voltage by a factor of two reduces the consumed energy by a factor of four. The dynamic energy ($E'(\psi_i)$) consumed by task ψ_i when executed at the reduced voltage level V_{level} [1] can be related to its nominal energy consumption, $E(\psi_i)$, as given below:

$$E'(\psi_i) = \left(\frac{V_{level}^2}{V_{supply}^2} \right) E(\psi_i) \Big|_{V = V_{supply}} \tag{1}$$

In this equation, it is seen that the larger the consumed energy by a task the larger the achieved energy reduction when scaling the supply voltage. So, the time schedule has to be modified to increase the available slack for tasks that have major influence on power/energy.

Reducing the body bias voltage yields an exponential leakage current/power reduction. This can be inferred from the leakage power equation given below:

$$P_{static} = V_{dd} K_3 e^{K_4 V_{dd}} e^{K_5 V_{bs}} + |V_{bs}| I_j \tag{2}$$

where V_{dd} is the supply voltage, V_{bs} is the body bias voltage, K_3, K_4, K_5, K_6, are constant fitting parameters which are technology dependent, and I_j is the current due to junction leakage [3].

As demonstrated above, voltage scaling causes significant power reduction, but scaling the voltage increases the circuit delay. The relation between path delay and voltage can be modeled similar to the alpha-model of an inverter as [3]:

$$d = \frac{L_d K_6}{(V_{dd} - V_{th})^{\alpha}} \tag{3}$$

In the above equation, α is a measure of the velocity saturation with typical value between *1* and *2*. L_d is the logic depth of the path. Reducing the supply voltage V_{dd} or

[1] V_{level} is less than (or equal to) the maximum voltage level V_{supply}

increasing the threshold voltage V_{th} increases the circuit delay. Therefore, available slack intervals directly influence the level to which the voltage can be reduced. Without any loss of generality, only DVS is applied here. Applying DVTS requires slight modifications, but the proposed algorithms are valid for both schemes.

4 Solving the Scheduling Problem

Scheduling in our case can be defined as optimizing the start time and the required voltage level(s) needed to execute each task. Therefore, the problem can be seen as a two dimensional (2-D) optimization problem: The start time $\tau_i(t)$ of each task ψ_i, and the voltage level(s) $V(\tau_i) = \{v_1, v_2, \ldots, v_n\}$ should be optimized to achieve maximum power/energy reduction. The time schedule is optimized in our approach with an eye on energy. Based on the time schedule the voltage schedule is optimized.

4.1 Time Scheduling

The time schedule in our approach has not only to guarantee time feasibility, but also to improve the availability and the distribution of the slack. A list based scheduler determines the start time of each task such that the time constraints are fulfilled and maximum possible energy reduction is achieved when applying voltage scaling. The fitness of each individual is evaluated after the 2-D scheduling process. The general time scheduling algorithm is depicted in Fig. (1).

```
Input: Tasks' Mapping, Technology Library
Output: Power-Aware Time Schedule

Get_PT(K_C);
Get_PP(K_C, Step);
C = 0;
REPEAT {
    FOR (i = 1) to N {
        Get(ST_C,i);
        Get(SP_C,i);
            IF(ST_C,i ≠ SP_C,i) {
                IF (ST_C,i can be delayed)
                    ST_C,i = SP_C,i;
            }
        Schedule_task (S, ST_C,i, C);
        Update (K_C,i);
        Get_PT(K_C,i);
        Get_PP(K_C,i);
    }
    C = C + 1;
    UNTIL (All tasks are scheduled)
}
```

Fig. 1. Time scheduling algorithm

The time scheduling algorithm computes the time priority for ready tasks, set K_C, by calling the function Get_PT(). The priority is determined initially based on ALAP and ASAP. The function Get_PP() ranks ready tasks according to their potential influence on energy reduction when applying voltage scaling. *Step* is the used time step as shown in Fig. 2. *N* is the number of allocated hardware components. Get($ST_{C,i}$) determines the set of ready tasks, $ST_{C,i} \subseteq K_{C,i}$, which has the maximum priority. Get($SP_{C,i}$) determines the set of ready tasks, $SP_{C,i} \subseteq K_{C,i}$, which has the highest power reduction effects when scheduled earlier in time. If the set $ST_{C,i}$ is different from the set $SP_{C,i}$, the priority of tasks with higher power rank is changed to be scheduled earlier if higher priority tasks can be delayed. The chosen tasks are scheduled at the given scheduling step using the function Schedule_task(). Subsequently, the set of ready tasks is updated for each hardware component by Update($K_{C,i}$).

The *power rank* (priority) is assigned to a tasks based on its potential influence on energy reduction. A task is scheduled in the scheduling step that leads to higher energy reduction according to equation 4. This in return increases the opportunity of tasks with high *power ranks* to utilize longer slack(s). Since scheduling tasks based on information related to the "critical path" only may cause other tasks to be scheduled near their deadlines. This may prevent additional energy saving when scaling the voltage. For instance, consider the case when tasks on the non-critical path have higher energy reduction effects. Delaying the execution of these tasks may lead to small or even no slack intervals available for these tasks.

We have experimentally observed that permitting ready tasks born at the k_{th} scheduling step to compete with tasks delayed from previous scheduling steps when ranking the tasks according to their energy reduction abilities causes inefficiency problems in some of the studied cases. To handle this deficiency, tasks can be delayed only *n* scheduling steps based on their power rank. This means that tasks with low power rank are scheduled after *n* scheduling steps depending on their time priority even if they can be delayed and their power rank is low. This is relatively significant to compensate for the lack of a lock-ahead mechanism in our fast heuristic in an attempt to avoid time-consuming methodologies. Due to this fast heuristics which avoids time-consuming look-ahead schemes, the resulting algorithm may not be optimal. But these cases where it may fail to yield remarkable improvements compared to general list scheduling are automatically eliminated by the general genetic optimization loop.

We further improved the efficiency of the proposed scheduling algorithm by executing it iteratively and one task is rescheduled at a time. The approach is based on the paradigm originally presented by Kernighan and Lin for graph partitioning [14]. According to ASAP and ALAP, a task is scheduled earlier or later as long as no data dependency or design constraint is violated. Each move leads to a new schedule. A task that is moved is then locked and is not allowed to move later unless all tasks are locked. The new (temporal) schedule is evaluated after scaling the voltage. After a set of *m* such moves, a subsequence of $q \leq m$ that maximizes the total power reduction can be reached. The schedule is then changed to include these *q* moves. The process is repeated until no further improvement is achieved.

A basic advantage in our algorithm is that it does not fail when the performance constraints are tight such as the one in [5]. The reason is that power and time aspects are considered at the same time in our scheduling approach.

4.2 Voltage Scheduling

The time schedule is adapted to increase the efficiency of voltage scaling techniques. The voltage levels required to execute the tasks should then be planned carefully in order to exploit the available slack intervals efficiently. This can achieve additional energy reduction when applying voltage scaling. Therefore, planning the voltage levels in our approach is based on a global view of all tasks and their energy profiles. The voltage scheduling algorithm is depicted in Fig. 2.

In this algorithm, y refers to the number of tasks. ΔEN_i refers to the energy saving for task (ψ_i) when extending its execution time (by one time step (Step, $n=1$)) by scaling its operating voltage:

$$\Delta EN_i = E(\psi_i) - E'(\psi_i)\Big|_{n=1} \tag{4}$$

The achieved energy reduction is related to ΔEN_i, according to equation (1). So, tasks with larger energy profiles are given more preference for extending their execution and reducing their energy. The power priority ($P_{priority}$) for task ψ_i is proportional to the calculated ΔEN_i multiplied by sl_i which is defined as:

$$sl_i = \begin{cases} 1, & slack_i \neq 0 \\ 0, & otherwise \end{cases} \tag{5}$$

The voltage scheduling algorithm terminates when one of two conditions is satisfied: 1) when the list LS is empty. This occurs if the voltage level is reduced to a value around $2V_{th}$ for all tasks. Actually, this voltage limit is chosen to avoid drastic increase in delay when voltage is reduced below this limit. 2) when $P_{priority} = 0$ for all tasks which means that there is no available slack to be exploited by any of the tasks.

Input: $F_p(\Psi,\Omega)$, $F_A(\Theta, \beta)$, *Mapping, Time Schedule, Step.*
Output: *Voltage Schedule* $V_{ss}(t)$

Step 0:
- Calculate ΔEN_i of all tasks $\psi_i \in \Psi$
- Assign $P_{priority}$ to all tasks $\psi_i \in \Psi$
- Create empty list LS of size y

Step 1:
Arrange the tasks in LS in a descending order of $P_{priority}$.

Step 2:
Get a (ψ_j) with the highest non-zero $P_{priority}$ from LS.
- If (V_{dd} is no longer $\geq 2V_T$) $\rightarrow$ remove ψ_j from LS.
- Else, extend the task (ψ_j) in steps of (n*$step$).
- Update the tasks profile.

Step 3:
Return if LS is empty OR all tasks have $P_{priority} = 0$

Step 4:
- Calculate ΔEN of all tasks in LS.
- Assign $P_{priority}$ to all tasks.
- Go to step 1.

Fig. 2. Planning the voltage levels planning

5 Experimental Results

A set of publicly available benchmarks was used to justify the applicability of our approach. The benchmarks include a set of 15 hypothetical examples generated originally by using the "Task Graphs For Free" TGFF [13]. Our 2-D scheduling methodology has been integrated to our automated co-synthesis tool. As a preliminary step, the general mapping and optimization approach presented in [11] have been implemented. In this preliminary step, we concentrated on optimizing the time schedule to satisfy performance constraints without tackling issues related to power. We call this step the 1-D scheduling.

The obtained energy reductions in percentage are presented in columns 2 and 3 of Table (1) for the 1- and 2-D scheduling, respectively. The last column presents the additional energy reductions achieved by the suggested 2-D scheduling algorithm over 1-D scheduling. The presented results show that all benchmarks made benefit of our 2-D scheduling algorithm, but in varying amounts.

The results can be categorized into two groups: The first group includes these benchmarks which made high benefit from the 2-D scheduling algorithm. This group of benchmarks indicates that for some applications the 1-D schedule might prevent the voltage scaling from achieving effective energy reductions. This reduces the efficiency of voltage scaling algorithms. The second group appears to make less benefit from our approach. This might partially be related to the performance constraints as well as to the mapping itself. At the same time, the lack of optimality in the algorithm might have an additional influence. But, in both cases, the 2-D scheduling algorithm is able to achieve additional energy reductions. Up to about 18% additional energy reduction was achieved by the 2-D scheduling algorithm over the 1-D. Additionally, the voltage planning algorithm which optimizes the voltage schedule partially hides the influence of time scheduling. Since the voltage scaling algorithm operates based on a global view of all tasks and their mappings.

Table 1. Achieved energy reductions by 1-D and 2-D scheduling

Benchmark	1-D Scheduling	2-D Scheduling	Extra Improvement
tgff1	68.1	83.5	15.4
tgff2	36.4	49.1	12.7
tgff3	64.6	75.9	11.3
tgff4	82.6	90.9	8.3
tgff5	60.1	61.1	1.0
tgff6	83.5	90.1	6.6
tgff7	30.2	45.0	14.8
tgff8	76.6	77.0	0.4
tgff9	37.3	44.9	7.6
tgff10	19.6	33.7	14.1
tgff11	24.3	33.6	9.3
tgff12	62.6	76.0	13.4
tgff13	60.9	73.2	12.3
tgff14	10.0	27.6	17.6
tgff15	14.1	27.9	13.8

6 Conclusions

An integrated methodology for low power/energy co-synthesis of real-time embedded systems is presented. The emphasis is on tasks' scheduling in time domain and scheduling the voltage level(s) required to execute each task. In this 2-D scheduling approach, the time schedule is prepared to increase the efficiency of voltage scaling algorithms while satisfying all performance constraints.

The presented results are encouraging and demonstrate that optimizing the time schedule increases the efficiency of voltage scaling algorithms. Additional energy reduction factor of up to about 18% is achieved when applying our 2-D scheduling methodology. The work also indicates the need to consider both power/energy consumption and performance constraints at the same time. This avoids time schedules which perform well in the time domain but causes low power/energy reductions when applying voltage scaling. It also avoids schedules that have high power/energy reduction effects but fail to satisfy performance constraints.

Acknowledgement

We are grateful indeed that DAAD (German Academic Exchange Service) supports this research since 2002.

References

1. Pering, T., Burd, T., Broderson, R.: Dynamic Voltage Scaling and the Design of a Low-Power Microprocessor System. In Power Driven Micro-Architectures Workshop, attached to ISCA'98, Barcelona, Spain, June, (1998).
2. Burd, T., Pering, T., Stratakos, A., Broderson, R.: A Dynamic Voltage Scaled Microprocessor System. IEEE Journal of Solid-State Circuits, vol. 35, no. 11, Nov. (2000) 1571-1580.
3. Martin, S., Flautner, K., Mudge, T., Blaauw, D.: Combined Dynamic Voltage Scaling and Adaptive Body Biasing for Lower Power Microprocessors under Dynamic Workloads. In International Conference on Computer-Aided Design, ICCAD'02, (2002) 721-725.
4. Pering, T., Burd, T., Broderson, R.: The Simulation and Evaluation of Dynamic Voltage Scaling. In International Symposium on Low Power Electronics and Design, ISLPED, (1998) 76-81.
5. Gruian, F., Kuchcinski, K.: LEnS: Task Scheduling for Low-Energy Systems Using Variable Supply Voltage Processors. In Asia and South Pacific Design Automation Conference, ASP-DAC, Jan. (2001) 449-455.
6. Bambha, N., Bhattacharyya, S., Teich, J., Zitzler. E.: Hybrid Global/Local Search for Dynamic Voltage Scaling in Embedded Multiprocessor. In the 1st International Symposium on Hardware/Software Co-Design, CODES'01, (2001) 243-248.
7. Ishihara, T., Yasuura, H.: Voltage Scheduling Problem for Dynamically Variable Voltage Processors. In International Symposium on Low Power Electronics and Design, ISLPED, (1998) 197-202.
8. Manzak, A., Chakrabarti, C.: Variable Voltage Task Scheduling for Minimizing Energy or Minimizing Power. In International Conference on Acoustics, Speech, and Signal Processing, Nov. (2000) 3239-3242.
9. Grajcar, M.: Genetic List Scheduling Algorithm for Scheduling and Allocation on a Loosely Coupled Heterogeneous Multiprocessor System. In the 36th ACM/IEEE Conference on Design Automation, (1999) 280-285.

10. Schmitz, M., Al-hashimi, B., Eles, P.: Energy-Efficient Mapping and Scheduling for DVS Enabled Distributed Embedded Systems. In Design, Automation and Test in Europe Conference and Exhibition, DATE, March, (2002) 514-521.
11. Schmitz, M., Al-Hashimi, B., Eles, P.: Synthesizing Energy-Efficient Embedded Systems with LOPOCOS. In Design Automation for Embedded Systems, (2002) 401-424.
12. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. Evolutionary Methods for Design, Optimization, and Control, CIMNE, Barcelona, Spain, (2002) 95-100.
13. Dick, R., Rhodes, D., Wolf, W.: TGFF: Tasks Graphs for Free. In International Workshop on Hardware/Software Codesign, March, (1998).
14. Kernighan, K., Lin, S.: An Efficient Heuristic Procedure for Partitioning Graph. Bell System Technical Journal, vol. 49, no. 2, (1970) 291-307.