Near Optimal and Energy-efficient Scheduling for Hard Real-time Embedded Systems**

Amjad Mohsen and Richard Hofmann

Department of Communication Systems and Computer Networks
University of Erlangen, Martensstr. 3, 91058 Erlangen, Germany.
Amjad.muhsen@infromatik.uni-erlangen.de, rhofmann@cs.fau.de

Abstract. In this paper, we present a new energy-aware scheduling scheme for real-time applications using architectures that employ voltage scaling technologies. Both dynamic voltage scaling (DVS) and dynamic threshold voltage scaling (DVTHS) can benefit from this scheduling scheme. The start time of each task is adapted to enhance the efficiency of voltage scaling schemes while still satisfying the required time feasibility. The introduced scheduling scheme is able to escape local minima and it can generate near-optimal schedules in terms of energy reduction. The scheduling paradigm is integrated into our automated and multiobjective system-level co-synthesis tool that performs system optimization. We report in this paper up to about 30% higher energy reduction compared to only performance-aware scheduling.

1 Introduction

The complexity of real-time embedded systems is increasing rapidly with a non-stopping demand for higher performance. Especially for mobile systems, power consumption is a real challenge since these systems depend on a battery that can supply energy only for limited time. The steady move towards deep submicron technologies will make power rather dominate all other design constraints. Therefore, new methodologies and automated tools are required to handle design tradeoffs at all abstraction levels in order to sharply reduce the consumed energy. Without such methodologies that deal with all requirements and tradeoffs concurrently, designed systems will continue to dissipate unnecessary energy that can be saved by using more efficient and energy-aware design tools.

A prominent approach which remarkably reduces total power/energy consumption is dynamic voltage scaling (DVS). DVS was proposed in order to trade performance for power without loosing the peak performance of the system [1]. Reducing the supply voltage by a factor of two in DVS-enabled systems reduces the consumed (dynamic) energy by a factor of four. As technologies continue to scale down, the leakage power will account to about 50% of the total consumed power or even more. Applying DVS alone can achieve only linear reduction in the leakage power. An

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

efficient scheme which can limit the leakage power is adaptive body biasing (ABB). In this scheme, the threshold voltage is modified based on demand according to the state of the system and its workload [2]. This scheme can cause exponential reduction in leakage power.

Applying voltage scaling schemes to exploit slack intervals in the system can reduce the total power/energy dissipated in digital systems. However, the availability of enough slack intervals and their distribution, which are directly influenced by the time schedule of the tasks, are crucial efficiency issues for voltage scaling schemes. Additionally, it is essential to consider power profiles of the tasks in distributed embedded systems when deriving the schedule [4]. Therefore, additional power/energy reduction can be achieved when the time schedule is planned with these issues in mind. The time schedule should be modified to enable tasks which are major consumers of energy to exploit longer slack intervals when applying voltage scaling.

This paper is organized as follows: The next section surveys some energy-aware related approaches. Section 3 reviews our energy-aware co-synthesis approach. Section 4 introduces basic concepts in voltage scaling schemes. In section 5, we present the proposed time scheduling algorithm. The obtained experimental results are presented and analyzed in section 6. We conclude this paper in section 7.

2 Related Work

Many approaches were proposed to solve the scheduling problem while considering energy issues. Gruian and Kuchcinski introduced a task scheduling heuristic based on list-scheduling [3]. The scheme dynamically calculated priorities of the tasks and tried to choose the best supply voltage levels (assuming DVS-enabled architectures) that minimize the consumed energy. The presented priority function was aware of energy aspect but might fail when the deadline was tight. To compensate for this drawback, a priority function tuning mechanism was used.

A static voltage scheduling problem was proposed and formulated as an integer linear programming (ILP) problem in [4]. The tasks were assumed to have different average switching capacitance per cycle to consider processing elements' power profiles. It was shown that considering the power profile when scaling the voltage could be a source of additional power reductions. Some other studies also reached a similar conclusion [5], [6]. However, these studies were done for single processor systems and did not show the effect of scheduling in the time domain on the performance of the voltage scheduler.

Grajcar suggested a genetic list scheduling algorithm without tackling the power problem [7]. In this approach, each individual represented an encoded schedule and the individuals of a population are ranked based on their fitness. A similar approach was used by Schmitz et al. in [8] to achieve energy–efficient scheduling when using DVS-enabled architectures. Priorities of the tasks were encoded into priority strings which are iteratively optimized by an evolutionary algorithm. The list scheduler then determines the start execution time based on the optimized priorities. The authors suggested the genetic list-based scheduling algorithm inside a global mapping optimization loop which was based on genetic algorithms.

Part of the proposed approaches may fail under stringent delay constraints. Others did not concretely tackle factors which have significant influence on total reduced energy such as continuous slack availability and distribution. We propose in this paper a near-optimum energy-aware scheduling scheme which modifies the time schedule iteratively in order to maximize the achieved energy reduction assuming that voltage scaling enabled architectures are used. The lengths of slack intervals available in the time schedule as well as their distribution are *improved* to achieve higher energy reduction. Different from previous approaches, our scheme has the ability to escape local minima. The scheduling algorithm is integrated inside our own automated genetic-based co-synthesis tool.

3 Overview of System-level Co-synthesis

System-level synthesis is considered here 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. In order to automate the synthesis process, system specification is captured into a system model.

3.1 System Model

The system is described here using two graphs: a task graph (*TG*), and an architecture graph (*AG*). Both graphs are automatically generated from SDL/MSC 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 precedence constraints and data dependencies $(\omega_i \in \Omega)$. Hard real-time constraint(s) T_i is/are forced on a node or a set of nodes as well as an absolute time constraint T_T. The *AG* is $F_A(\Theta,\Re)$, where Θ stands for the set of available architectures $(\theta_i \in \Theta)$ and $(\rho_i \in \Re)$ represents possible connections between hardware components. For each 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 power-, delay-, and cost-scaling when using this type.

3.2 Optimization Approach

System-level synthesis requires optimizing three basic sub-problems: First, allocation (α) which selects a set of hardware components for implementation purposes. Second, mapping (β) which determines which single hardware resource is used to execute each task. Based on the allocation and mapping, the schedule of tasks is then optimized to achieve time feasibility and to maximize energy saving when applying voltage scaling. In our approach, a voltage schedule specifies the voltage levels required to execute each task. Therefore, the schedule (sc) includes a time and a voltage schedule. The time schedule is optimized to achieve time feasibility and to maximize simultaneously energy reduction when applying voltage scaling.

The synthesis process is considered in our approach as a multiobjective optimization problem which can be stated as: *minimize the function* $f(\alpha,\beta,sc)$ = ($f_1(\alpha,\beta,sc)$, $f_2(\alpha,\beta,sc)$, $f_3(\alpha,\beta,sc)$) while all design constraints are satisfied. This formulation considers three objectives: performance (f_1), power (f_2), and cost (f_3). The proposed synthesis methodology applies evolutionary algorithms (EAs) to optimize the allocation/binding (global optimization) based on the objective vectors (f_1, f_2, f_3). The schedule is optimized for each individual separately as an integrated part of the global optimization loop. EAs are suitable for such optimization problems since they work in parallel on a population of individuals. Each individual encodes a potential implementation. Also, EAs work well on problems with large search spaces.

The EA consists of an optimization loop inside which the principles of reproduction, crossover and mutation are applied on the population's individuals. The purpose is to find iteratively better population based on a set of design objectives. Each individual is evaluated and assigned a fitness value based on the objective vector and fittest individuals are allowed to mate. Fitness values reflect the superiority of each individual in terms of *all* objectives. Computing fitness and performing selection are carried out by the widely used evolutionary optimizer, SPEA2 which is integrated to our synthesis tool. SPEA2 is an up-to-date optimizer which offers several merits over normal evolutionary approaches [9]. The optimization loop terminates when the pre-specified condition(s) is/are satisfied. A set of implementation alternatives which spans the Pareto-optimal front are produced. Each implementation alternative includes an allocation, a mapping, and a time- and voltage-schedule. The designer selects one of these design alternatives based on market requirements.

4 Voltage Scaling

Because of its quadratic relationship to power, voltage reduction has the most drastic means of reducing total consumed power/energy. DVS increases the energy efficiency of a device by dynamically adapting its speed and voltage as needed. However, in heterogeneous systems, the variation in the power profiles of the tasks when executed on different components influences the amount of energy reduction that can be achieved when scaling the voltage. This can be seen in the equation below:

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

In the above equation, $E'(\psi_i)$ refers to the energy consumed by task ψ_i after scaling the operating voltage to V_{level} ($V_{level} \leq V_{supply}$). $E(\psi_i)$ stands for the energy consumed by this task at nominal voltage (V_{supply}) based on a certain mapping. Therefore, the higher the consumed power and the lower the adopted voltage level to execute a task the more energy reduction is achieved.

Voltage scaling schemes can also be applied to limit the amount of consumed leakage power in digital circuits. Dynamic threshold voltage (V_{th}) scaling (DVTHS) is a candidate scheme which dynamically adapts V_{th} by means of varying the body bias

voltage (V_{bs}) in order to save leakage power. The relationship between V_{bs} and the consumed static power (P_{static}) is given below:

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

In this equation, V_{dd} is the supply voltage, K_3, K_4, K_5, are constant fitting parameters which are technology dependent, and I_j is the current due to junction leakage [2].

Voltage scaling is an efficient method to reduce the total consumed power/energy. Nevertheless, scaling the voltage has negative influence on circuit delay which can be formulated as follows [2]:

$$D = \frac{K_6 L_d V_{dd}}{((1+K_1)V_{dd} + K_2 V_{bs} V_{bs} - V_{th1})^{\alpha}} \qquad (3)$$

In this equation, K_1, K_2, K_6, and V_{th1} are circuit dependent constants. L_d is the logic depth and α is the logic depth which is technology dependent ($1.4 \leq \alpha \leq 2$). Hence, voltage scaling mechanisms can only be applied if degradation in performance can be tolerated. Key efficiency factors for voltage scaling schemes are the availability of enough continuous slack intervals and the way in which these intervals are exploited. The proposed scheduling scheme which is presented in the next section handles these issues efficiently. It constructs a time schedule which considers future needs of voltage scaling schemes. Without any loss of generality, we examined the proposed algorithms by applying DVS only.

5 Energy-Efficient Time Scheduling

Scheduling in our case can be defined as determining the start time and the required voltage level(s) needed to execute each task. Precedence constraints and data dependencies must be fulfilled. Hence, the problem can be seen here as a two dimensional (2-D) optimization problem. The start time of each task ψ_i, $\tau_i(t)$, and the voltage level(s) $V(\tau) = \{v_1, v_2, ..., v_n\}$ should be optimized to maximize the power/energy reduction. In our approach, the time schedule is determined first with an *eye on energy*. Based on the derived time schedule, the voltage schedule is determined based on a global view of all tasks. So, the objective is to optimize the start execution time of each task such that more energy reduction can be achieved when applying voltage scaling schemes. In this section, the time scheduling is presented whereas the proposed voltage scheduling algorithm can be found in [10].

As mentioned previously, the available slack intervals and their distribution among the tasks are basic efficiency factors for voltage scaling schemes. Considering power profiles of different processing elements when executing different tasks makes the influence of slack distribution more crucial. Therefore, the time schedule has to be adapted such that tasks which are major consumers of energy exploit longer slack intervals. Moreover, tasks which consume more energy should have higher priority to scale their operating voltage when deriving the voltage schedule. The proposed scheduling algorithm considers all these issues as explained below.

5.1 Scheduling Algorithm

The proposed time scheduling algorithm considers *future* needs of voltage scaling schemes in order to maximize energy reduction. The algorithm has an iterative nature and it is based on the paradigm originally proposed by Kernighan and Lin (KL) for graph partitioning problem (min-cut partitioning) [11]. Based on ASAP and ALAP schedules, and the mobility value of each task, initial priorities are computed for all tasks. A list-based scheduler is then used to generate an initial feasible schedule. According to ASAP and ALAP, a task can be scheduled earlier or later as long as no data dependency or delay constraint will be violated. For example, task ψ_8 can be scheduled between 12 and 22 in Fig. 3.

The proposed scheduling algorithm iteratively selects a task and a scheduling step based on the ready list of an allocated hardware such that no delay constraint will be violated. The scheduling step that maximizes the energy reduction (when applying voltage scaling) is fixed for this task and the task itself is locked and disallowed to move later unless all tasks are locked. One task is moved at a time and each move leads to a new schedule. The new (*temporal*) schedule is repeatedly evaluated after scaling the voltage. After a set of m such moves, a subsequence of $q \leq m$ that maximizes the total energy reduction can be reached. The schedule is then changed to include these q moves. The process is repeated and all tasks are unlocked until no further improvement in terms of energy could be achieved. This scheduling approach has been chosen because KL is known to find good solutions in small CPU time and it has been successfully applied for long time in various placement and routing applications. At the same time, this methodology allows *bad* solutions to arise and then chooses the set of moves that leads to the maximum energy reduction.

The entire scheduling algorithm is shown in details in Fig. 1. In this algorithm, Work_sched and Temp_sched are temporary schedules whereas UpToDate_sched represents the maintained up-to-date schedule. Power_gain[] stores the power gain for each iteration. The function MOVE_TASK(Work_sched, task(i), step$_{j,r}$) returns a new schedule after modifying Work_sched by scheduling task(i) into step$_{j,r}$; j represents a scheduling step and r stands for the selected ready list. The *function* POWER() takes a schedule and computes its power/energy consumption. The two arrays Sequence[] and Sched_step[] maintain the sequence of moved tasks and the corresponding scheduling steps, respectively.

The iterative nature of the algorithm might appear to be a disadvantage. When all possible moves for each task are considered the time complexity will be $O(n)$, where n is the total number of tasks and moves. At the same time, implementing DVS requires substantial algorithms to determine the required operating voltage and the corresponding speed/frequency. These algorithms affect the performance of the device and increase the overall consumed energy. In our approach, we emphasize that scheduling processes are off-line processes that optimize start time and voltage level(s) required to execute each task during the design phase. The planned voltage levels and the time schedule are stored in a table form for run-time use. As a result, the additional energy consumption related to the scheduling process is omitted and the run-time performance overhead due to this additional process is reduced to a minimum. However, voltage switching cost in terms of energy is still considered during run-time.

```
Input: Fp(Ψ,Ω), FA(Θ, Ϟ), Mapping
Output: Time Schedule: UpToDate_sched

REPEAT
   Unlock all tasks;
   Work_sched = UpToDate_sched;
   C = 0;
   WHILE There is a movable task
      C = C + 1;
       Power_gain[C]  = - ∞ ;
      FOR each task(i) ∈ {movable tasks}
         FOR each scheduling step in the ready list (step_j,r)
            Temp_sched = MOVE_TASK(Work_sched, task(i), step_j,r);
            Temp_gain = POWER(Work_sched) – POWER(Temp_sched);
            IF Temp_gain > Power_gain[C];
                 Power_gain [C] =  Temp_gain;
                 Sequence [C] = task(i);
                 Sched_step[C] = step_j,r;
            ENDIF
         ENDFOR
      ENDFOR
      Lock (moved tasks)
      Work_sched=MOVE_TASK(Work_sched, Sequence [C], Sched_step[C]);
      ENDWHILE
      Select a sequence q based on Power_gain[];
      IF maximum cumulative power reduction (Po_R)> 0
         Update the UpToDate_sched with the sequence;
      ENDIF
UNTIL Po  R < 0;
```

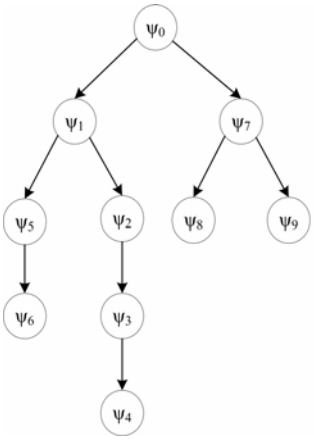
Fig. 1. Energy-aware time scheduling algorithm

5.2 Illustration Example

To demonstrate the influence of the proposed energy-ware time schedule consider the task graph presented in Fig. 2. Tasks' mapping and other supporting information regarding worst case execution time (WCET) and power consumption are found in the table shown on the right hand side in the same figure. Fig. 3 shows an initial time schedule based on the information presented in Fig. 2. The time schedule is derived using a list-based scheduling algorithm and it is presented here as a Gantt chart.

Columns 3 and 4 in the table shown in Fig. 1 give the WCET and the average power consumption for each task according to the given mapping, respectively. The start time for each task according to "as soon as possible" (ASAP) and "as late as possible" (ALAP) scheduling algorithms are given in columns 5 and 6, respectively. The mobility which is defined in column 7 of the same table is defined as the

difference between ALAP and ASAP. The priority of each task is calculated based on the mobility. Based on the table shown in the above figure, it can be seen that tasks which have major energy reduction effects (high power profiles), such as tasks ψ_6, ψ_8, and ψ_9, have low priority. This means that they are scheduled in time after other higher priority tasks. As it can be seen in Fig. 3, task ψ_6 has no slack at all while tasks ψ_8 and ψ_9 have both 6 slack slots assuming that the deadline is 26. No other task can make any benefit from voltage scaling schemes.



Task	Mapping	WCET	Power	ASAP	ALAP	Mobility
ψ_0	HW1	2	10	0	9	9
ψ_1	HW1	5	15	2	11	9
ψ_2	HW2	4	10	7	16	9
ψ_3	HW2	2	5	11	20	9
ψ_4	HW2	4	10	13	22	9
ψ_5	HW2	4	10	7	17	10
ψ_6	HW1	5	25	11	21	10
ψ_7	HW1	5	15	2	17	15
ψ_8	HW0	4	50	7	22	15
ψ_9	HW0	4	50	7	22	15

Fig. 2. Task graph and mapping example

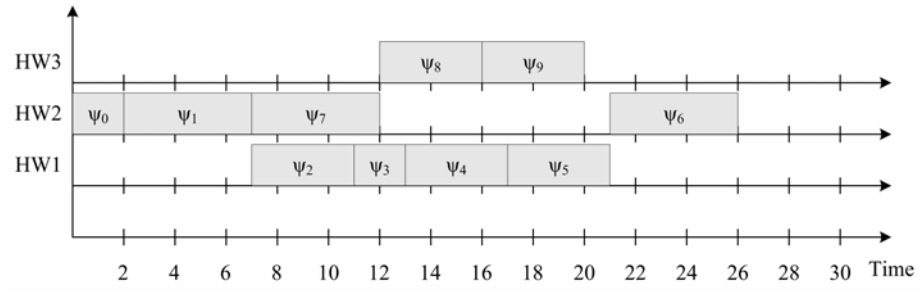


Fig. 3. Initial schedule

Consider now the time schedule in Fig. 4 where task ψ_5 is scheduled before task ψ_4, independent on their priorities. As it can be seen in the figure, additional 4 slack slots are now available for task ψ_6 which causes higher energy reduction when applying voltage scaling. A more efficient time schedule from the energy point of view is shown in Fig. 5. In this figure, 9 time slots are now available to tasks ψ_8 and ψ_9. Task ψ_6 has now 5 time slots which can be exploited when applying voltage

scaling. In this schedule ψ_7, for example, is scheduled before ψ_1 although the later has higher priority.

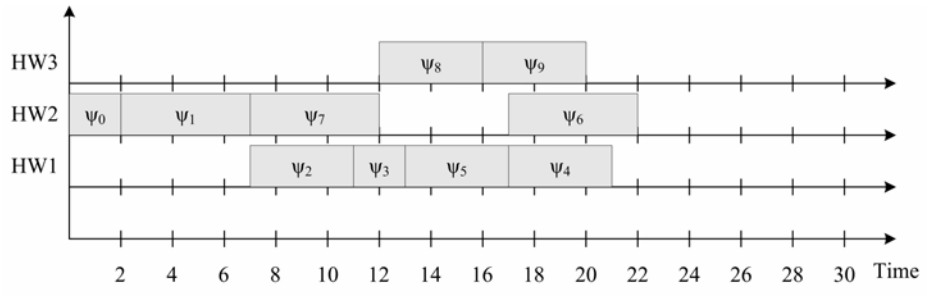


Fig. 4. Modified schedule

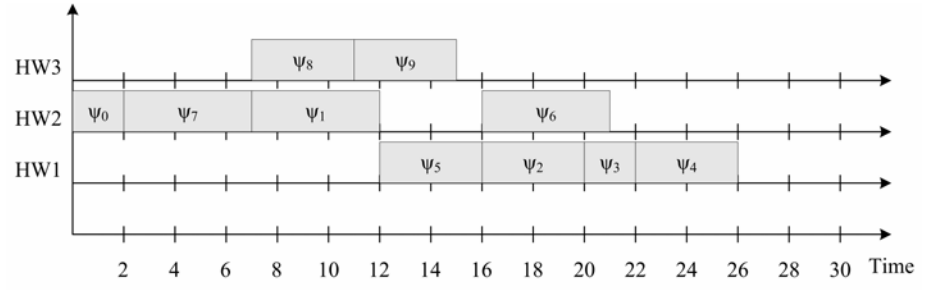


Fig. 5. An energy-efficient time schedule

6 Experimental Results

A set of benchmarks are used to prove the applicability and the efficiency of our approach in reducing the total consumed energy. The benchmarks include a set of 20 publicly available benchmarks which are generated originally by using "Task Graphs For Free" TGFF [12] and obtained from [8].

Our 2-D scheduling methodology has been integrated to our automated co-synthesis tool which is presented previously. For comparison purposes, another energy-efficient time scheduling algorithm was also implemented and integrated to our tool. This later time scheduling algorithm is called evolutionary list-based scheduling algorithm (ELSA). This algorithm is also based on a list scheduling algorithm. Priorities of tasks are optimized using an up-to-date evolutionary algorithm. An optimization objective is increasing the energy reduction when applying voltage scaling. Based on the *optimized* priorities, the list schedule determines the start execution time for each task. An energy-ware scheduling algorithm similar to ELSA was proposed in [8].

In order to enable fair comparison, the 2-D and ELSA schemes were tested under the same conditions. To achieve this, we implemented ELSA and integrated it to our global optimization tool. At the same time, the same voltage scheduling algorithm was applied in all experiments. To further demonstrate the influence of energy-aware scheduling, another separate experiment was conducted. In this experiment, a traditional list scheduling algorithm was implemented to optimize the time scheduling without considering energy. We call this scheme the 1-D scheduling. Results of this experiment are shown in columns 2 and 6 of Table 1. Columns 3 and 7 present the energy reduction achieved when applying ELSA whereas energy reductions obtained when applying the 2-D scheme are shown in columns 4 and 8 for all the included benchmarks. Obtained energy reductions are reported here in percentage.

Table 1. Energy reduction in % obtained when using 1-D, ELSA, and 2-D scheduling

Benchmark	1-D	ELSA	2-D	Benchmark	1-D	ELSA	2-D
tgff1	68.1	82.6	83.5	tgff11	24.3	30.1	33.6
tgff2	36.4	43.9	49.1	tgff12	62.6	74.6	76.0
tgff3	64.6	70.8	75.9	tgff13	60.9	72.0	73.2
tgff4	82.6	88.0	90.1	tgff14	10.0	26.4	27.6
tgff5	60.1	61.1	61.1	tgff15	14.1	27.4	27.9
tgff6	83.5	87.8	90.1	tgff16	27.4	37.9	39.1
tgff7	30.2	43.3	45.0	tgff17	40.0	58.8	63.1
tgff8	76.6	76.7	77.0	tgff18	31.0	43.5	43.6
tgff9	37.3	38.1	45.0	tgff19	47.0	58.3	76.7
tgff10	19.6	31.7	33.7	tgff20	78.5	84.1	86.6

The above presented results show that all benchmarks made benefit of the 2-D scheduling scheme, but in varying degrees. The maximum energy reduction (90.1%) is obtained for tgff4 and tgff6. The presented results indicate that the 2-D scheduling scheme can perform at least as good as the ELSA algorithm. Additional energy reduction of up to 18.4% could be achieved by the 2-D scheduling over ELSA. This can be related to fact that the 2-D scheduling scheme can escape local minima. It accepts unacceptable moves for the tasks as long as these belong to a scheduling sequence that maximizes the overall energy reduction. The results obtained when applying the 2-D scheme shows that up to about 30% higher energy reduction can be achieved over 1-D scheduling (TGFF19). ELSA can only achieve up to about 19% higher energy reduction compared to the 1-D scheduling scheme (tgff17).

7 Summary and Conclusion

This paper presented a new two-dimensional (2-D) scheduling scheme that is able to remarkably increase energy reduction obtained when applying voltage scaling. This

scheduling scheme has the tendency to reach near-optimal schedules in terms of energy when applying voltage scaling. This is achieved by adopting the set of task-moves that lead to a maximum *cumulative* energy reduction which enables the algorithm to escape local minima. Experimental results indicate that up to about 30% energy reduction could be achieved.

A basic conclusion that can be drawn based on the experimental results is that it is essential to consider energy-related issues when optimizing the time schedule. Time feasibility should firstly be guaranteed but at the same time the schedule should be adapted based on certain optimization criterion to increase its energy-efficiency.

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. Martin, S., Flautner, K., Mudge, T., Blaauw, D.: Combined Dynamic Voltage Scaling and Adaptive Body Biasing for Lower Power Microprocessors under Dynamic Workloads. In *Proceedings of the International Conference on Computer-Aided Design*, ICCAD'02, (2002) 721-725.
3. Gruian, F., Kuchcinski, K., LEnS: Task Scheduling for Low-Energy Systems Using Variable Supply Voltage Processors. In *Proceedings of Asia and South Pacific Design Automation Conference*, ASP-DAC, January (2001) 449-455.
4. Ishihara T., Yasuura, H.: Voltage Scheduling Problem for Dynamically Variable Voltage Processors. In *Proceedings of the International Symposium on Low Power Electronics and Design*, ISLPED, (1998) 197-202.
5. Okuma, T., Ishihara, T., Yasuura, H.: Real-Time Task Scheduling for a Variable Voltage Processor. In *Proceedings of the 12th International Symposium on System Synthesis*, ISSS, (1999) 24-29.
6. Manzak A., Chakrabarti, C.: Variable Voltage Task Scheduling for Minimizing Energy or Minimizing Power. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, November (2000) 3239-3242.
7. Grajcar, M.: Genetic List Scheduling Algorithm for Scheduling and Allocation on a Loosely Coupled Heterogeneous Multiprocessor System. In *Proceedings of the 36th ACM/IEEE Conference on Design Automation*, (1999) 280-285.
8. Schmitz, M., Al-hashimi, B., Eles, P.: Energy-Efficient Mapping and Scheduling for DVS Enabled Distributed Embedded Systems. In *Proceedings of Design, Automation and Test in Europe Conference and Exhibition, DATE,* March, (2002) 514-521.
9. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In *Proceedings of Evolutionary Methods for Design, Optimization, and Control, CIMNE,* Barcelona, Spain, (2002) 95-100.
10. Mohsen, A., Hofmann R.: Efficient Voltage Scheduling and Energy-aware Co-synthesis for Real-time Embedded Systems. Tenth Asia-Pacific Computer Systems Architecture Conference, Singapore, October 24-26, 2005.
11. Kernighan K., Lin, S.: An Efficient Heuristic Procedure for Partitioning Graph. *Bell System Technical Journal*, vol. 49, no. 2, (1970) 291-307.
12. Dick, R., Rhodes, D., Wolf, W.: TGFF: Tasks Graphs for Free. In *Proceedings of International Workshop on Hardware/Software Codesign*, March, (1998).