Considering Power Variations of DVS Processing Elements for Energy Minimisation in Distributed Systems

Marcus T. Schmitz
Dept. of Electronics and Computer Science
University of Southampton
Southampton, SO17 1BJ, United Kingdom
m.schmitz@ecs.soton.ac.uk

Bashir M. Al-Hashimi
Dept. of Electronics and Computer Science
University of Southampton
Southampton, SO17 1BJ, United Kingdom
bmah@ecs.soton.ac.uk

ABSTRACT

Dynamic voltage scaling (DVS) is a powerful technique to reduce power dissipation in embedded systems. Some efficient DVS algorithms have been recently proposed for the energy reduction in distributed system. However, they achieve the energy savings solely by scaling the system task with respect to the timing constraints, while neglecting that power varies among the tasks executed by DVS processing elements (DVS-PEs). In this paper we investigate the problem of considering DVS-PE power variations dependent on the executed tasks, during the synthesis of distributed embedded systems and its impact on the energy savings. Unlike previous approaches, which minimise the energy consumption by exploiting the available slack time without considering the PE power profiles, a new and fast heuristic for the voltage scaling problem is proposed, which improves the voltage selection for each task dependent on the individual power dissipation caused by that task. Experimental results show that energy reductions with up to 80.7% are achieved by integrating the proposed DVS algorithm, which considers the PE power profiles, into the co-synthesis of distributed systems.

1. INTRODUCTION

Power dissipation has become a major issue in embedded system design, due to reliability and battery life time in portable application. One technique to tackle the problem of power consumption is dynamic voltage scaling [19], which reduces the power dissipation P by dynamically scaling the supply voltage V_{dd} and operational frequency f of the embedded processor [3, 10, 14] according to the system requirements. The power dissipation and the operational frequency are expressed by the following equations:

$$P_{dyn} = C_L \cdot N_{0\to1} \cdot f \cdot V_{dd}^2 \quad (1)$$

$$f = k \cdot (V_{dd} - V_t)^2 / V_{dd} \quad (2)$$

ISSS'01, October 1-3, 2001, Montréal, Québec, Canada.

where C_L is the switching capacitance, $N_{0\to1}$ denotes the switching activity, k is a circuit dependent constant and V_t is the circuit threshold voltage.

There has been extensive research on DVS algorithms with early research focused on single processor systems executing independent tasks [11, 21]. Recently DVS has also been applied to distributed embedded systems. In [9] a task scheduling algorithm based on a list scheduling with dynamic re-calculation of priorities is presented. The algorithm attempts to find optimised and feasible schedules by assigning priorities based on the average energy dissipation. If no feasible schedule is found the priorities of the tasks on the critical path are increased and the tasks are rescheduled. The scheduling algorithm presented in [16] aims at a power-conscious joint scheduling of periodic task graphs and aperiodic tasks, which employs DVS and dynamic power management to achieve energy efficiency. They optimise a pre-ordered schedule by distributing the flexibility in the schedule more evenly. These algorithms show that DVS can be applied to distributed embedded systems.

In all of the reported scaling algorithms for distributed systems [9, 16] the PE power consumption of different tasks is neglected during the voltage selection. However, modern VLSI designs make extensive use of low power techniques like, e.g., gated clocks [5], and the PEs in distributed systems are often of heterogeneous nature [20]. Certainly, this results in a PE power profiles, which show power variations. In [13, 17, 18] it was shown that the consideration of PE power variations of the executed task is important from the energy reduction point of view. They proposed suitable DVS algorithms for single DVS-PE systems. However, the trend of embedded systems implementations is going towards distributed systems [20].

The aim of this paper is to investigate the problem of considering DVS processor power variations dependent on tasks, during the synthesis of distributed embedded systems and its impact on energy savings. A new off-line scaling heuristic is proposed, which is sufficiently fast enough (polynomial time complexity) to permit its use during the design space exploration phase of the co-synthesis process, further reducing the energy dissipation by optimising the scheduling and mapping of the specification tasks towards DVS usability.

The remainder of the paper is organised as follows. Section 2 motivates the consideration of PE power variations among tasks, using an illustrative example. In Section 3 the specification model is given and the variable power DVS problem is formulated. Section 4 presents a constructive

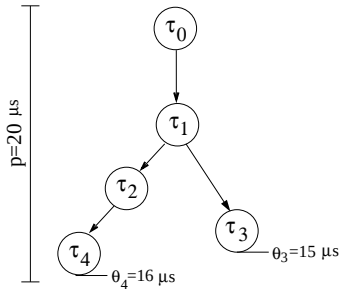


Figure 1: Task graph example

task	*WCET* (μs)	*Power* (mW)	*Mapping*
τ_0	1.5	85	PE0
τ_1	3.0	20	PE1
τ_2	7.5	15	PE1
τ_3	1.5	80	PE1
τ_4	1.5	100	PE0
$\tau_0 \rightarrow \tau_1$	0.5	5	CL0
$\tau_3 \rightarrow \tau_4$	1.0	5	CL0

Table 1: Exaction times, power dissipations, and mapping of tasks and communications

heuristic to solve the variable power DVS problem. In Section 5 experimental results are given, which demonstrate the necessity of considering PE power variations. Section 6 presents some conclusions.

2. MOTIVATIONAL EXAMPLE

This section addresses why PE power variations among tasks in distributed systems need to be considered in order to reduce the system energy consumption. This is illustrated by the following simple example. Fig. 1 gives a system specification using a task graph (TG), with two hard deadlines at $15\mu s$ and $16\mu s$. The period between two invocations of the task graph is $20\mu s$. The worst-case execution times (WCET), the nominal power dissipations and the mapping of each task and communication are shown in Table 1.

Scheduling the given specification on a distributed system consisting of two PEs (PE0 and PE1) connected by a link (CL0), could result in the feasible schedule (executing tasks at nominal supply voltage) given in Fig. 2. Using the execution values in Table 1 the dynamic energy dissipation of the system can be calculated as $577.5nJ$. Obviously, task τ_3 and τ_4 finish execution before their deadlines are exceeded, leaving a slack of $1\mu s$, which can be used to reduce the energy dissipation through supply voltage scaling and hence reducing the power dissipated by the tasks. Lets consider two cases for the identification of scaling voltages: First, when the voltage selection process neglects the individual PE power dissipation and second, when the power profile is taken into account in order to refine the voltage schedule.

One approach to optimise the energy dissipation, which neglects the power profile, is to distribute the slack time evenly among the tasks (similar to [16]), as shown in Fig. 3(a). Each task execution in this example is extended by a factor of $14.5/13.5 = 1.074$, since the total execution times (both path of the TG) account for $13.5\mu s$ and the slack is $1\mu s$. Thereby the new execution times can be calculated as

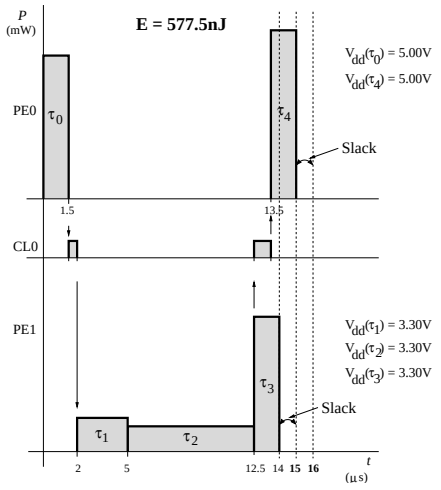


Figure 2: Tasks execution at nominal supply voltage

$t_0 = 1.61\mu s$, $t_1 = 3.22\mu s$, $t_2 = 8.56\mu s$, $t_3 = 1.61\mu s$, and $t_4 = 1.61\mu s$. According to this execution times the supply voltages of PE0 and PE1 can be reduced to $4.79V$ and $3.16V$, respectively, using Equation (2). This values were calculate taking into account the different maximal supply voltages V_{max} and threshold voltages V_t of PE0 ($V_{max} = 5V, V_t = 1.2V$) and PE1 ($V_{max} = 3.3V, V_t = 0.8V$). Using the new execution times, the individual task power dissipations can be reduced to $P_0 = 72.57mW$, $P1 = 17.08mW$, $P_2 = 12.81mW$, $P_3 = 85.42mW$, and $P_4 = 85.38mW$, as calculated by Equation (1). This results in a total energy dissipation $E = 530.3nJ$, a reduction of 8.2%.

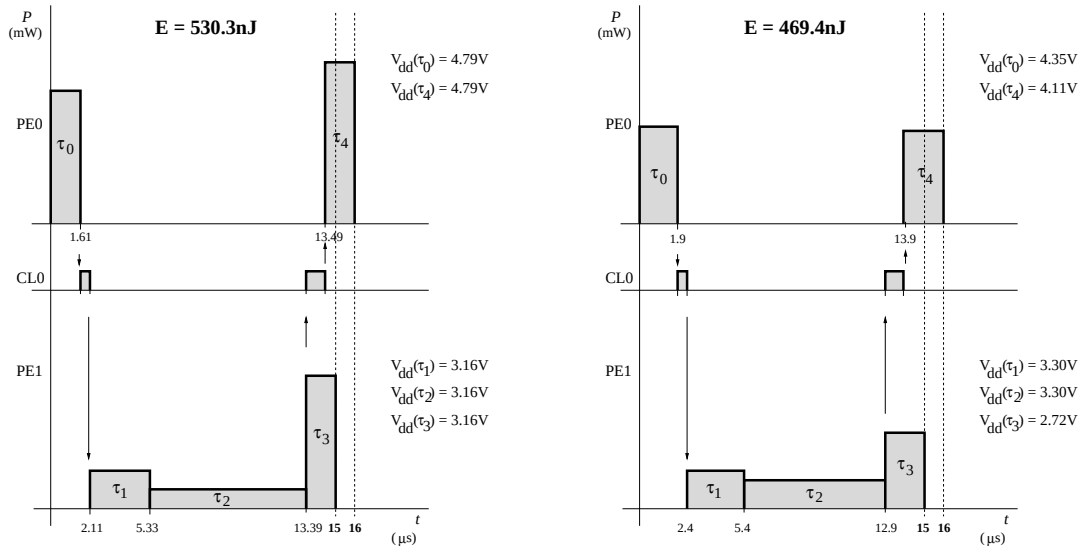
Now consider the second case when the PE power dissipation dependent on tasks is considered during the voltage selection. Intuitively, the tasks τ_0,τ_3 and τ_4 will reduces the energy dissipation more than the tasks τ_1 and τ_2, when voltage scaled, due to their high power consumption and therefore should be extended more then the remaining tasks. Fig. 3(b) shows an improved voltage schedule. This schedule was generated using the scaling algorithm introduced later in Section 4. To demonstrate the principle behind the proposed approach it is necessary to give the following definition.

DEFINITION 1. *We define an energy gradient Δt_τ as the difference between the energy dissipation of task τ with the execution time t and the reduced energy dissipation (due to voltage and clock scaling) of the same task, when extended by a time quantum Δt. Formally:*

$$\Delta E_\tau = E_\tau(t) - E_\tau(t + \Delta t) \tag{3}$$

where $E_\tau(t)$ and $E_\tau(t + \Delta t)$ are calculated using Equations (1) and (2). □

For a simple illustration of the proposed method the time quantum Δt is set to $0.1\mu s$. Based on this time quantum an energy gradient ΔE_τ can be calculated for all tasks, using Equation (3). The resulting energy gradients are $\Delta E_0 = 127.5nJ - 117.9nJ = 9.60nJ$, $\Delta E_1 = 2.34nJ$, $\Delta E_2 = 1.56nJ$, $\Delta E_3 = 8.99nJ$, and $\Delta E_4 = 11.3nJ$. Obviously, the task with the highest energy gradient ΔE_τ will improve the total energy dissipation by the highest amount and therefore this task is extended by Δt; in our case task τ_4. Due to the



(a) Voltage scheduling assuming constant power of tasks

(b) Voltage scheduling of the proposed approach allowing for power variations

Figure 3: Voltage schedules showing reduced energy consumption

non-linear relationship of execution time and power dissipation, the energy gradient needs to be re-calculate after each extention of a task. This is $\Delta E_4 = 9.65nJ$ for task τ_4 in the first iteration. Based on these observations, tasks with the highest energy gradient are iteratively extended until no further extention is possible due to hard deadline constraints. This results in the schedule shown in Fig. 3(b), which reveals a total energy dissipation of $469.4nJ$, a reduction of 18.7%. Although using a simple example, this clearly indicates the advantage of voltage scaling methods, which take the power profile into account during supply voltage optimisation.

3. MODEL SPECIFICATION AND PROBLEM FORMULATION

Specifications of typical embedded systems are often represented by a task graph model. These are directed acyclic graphs $G_S(\mathcal{T}, \mathcal{C})$ where $\mathcal{T} = \{\tau_0, \tau_1, \ldots, \tau_N\}$ represent tasks, $\mathcal{C} = \{\gamma_{ij} = (\tau_i, \tau_j, \chi_{ij})\}$ represent precedence constraints and data dependencies (also called communications) between two tasks (τ_i, τ_j), and χ_{ij} indicates the data amount to be transfered. Tasks might be associated with a hard deadline θ_i by which its execution must be finished, in order to fulfil feasibility requirements. The target architecture, represented by a directed graph $G_A(\mathcal{P}, \mathcal{L})$, consists of processing elements $\pi \in \mathcal{P}$ (e.g. general purpose processors, ASIPs, DVS-μPs, etc.) and communication links (CL) $\lambda \in \mathcal{L}$ (e.g. point-to-point connections, busses, etc.). For each task/PE and communication/CL combination, properties like execution time and power dissipation are given beforehand. These values are either based on previous design experience or on sophisticated estimation techniques [2, 15].

3.1 Power Variation DVS Problem

Unlike previous approaches, which identify supply voltages for each task execution without the consideration of the dissipated power, our optimisation problem involves a refined voltage selection for each task, dependent on the individual power dissipation caused by task execution. The problem can be stated as followed:

> Find for all DVS-PE mapped tasks $\tau \in \mathcal{T}_{\mathrm{DVS}}$ of the system specification a single scaling voltage V_{dd} (between the threshold voltage V_t and the nominal supply voltage V_{max}), under consideration of the individual power dissipations $P_{max}(\tau)$ and worst case execution time $t_{min}(\tau)$, such that the dynamic energy dissipation E_Σ is minimised and no deadline and precedence constraints are violated.

Therefore, minimise

$$E_\Sigma = \sum_{\tau \in \mathcal{T}_{\mathcal{DVS}}} \frac{P_{max}(\tau) \cdot t_{min}(\tau)}{V_{max}^2(\tau)} \cdot V_{dd}^2(\tau) \tag{4}$$

subject to

$$V_t(\tau) < V_{dd}(\tau) \leq V_{max}(\tau), \forall \tau \in \mathcal{T}_{DVS} \tag{5}$$

limiting the applicable supply voltages range. Further, no hard deadline violation can be tolerated in order to yield feasibility and therefore

$$t_S(\tau) + t_{exe}(\tau) \leq \theta(\tau), \forall \tau \in \mathcal{T}_d \tag{6}$$

where t_S denotes the task start time and $\mathcal{T}_d$ is the set of all hard deadline tasks. The execution time function $t_{exe} : \mathcal{T} \rightarrow \mathbb{R}^+$ follows equation $(t_{min}(\tau) \cdot V_{dd}(\tau) \cdot (V_{max}(\tau) - V_t(\tau)^2))/((V_{dd}(\tau) - V_t(\tau))^2 \cdot V_{max}(\tau))$, which can be derived from Equation (2). Furthermore, the execution order of the tasks and communications must be respected and is expressed as follows:

$$t_S(\gamma) + t_{com}(\gamma) \leq t_S(\tau), \forall \tau \in \mathcal{T}, \gamma \in \mathcal{C}^{in}(\tau) \tag{7}$$

where t_{com} presents the communication time of communication event γ and $\mathcal{C}^{in}$ is the set of all ingoing edges of task τ. In addition, no execution overlap on any resource $\epsilon \in (\mathcal{P} \cup \mathcal{L})$ is permitted and therefore the execution intervals i, formed by the start and end times of events, are not allowed to intersect, as given formally in Equation (8).

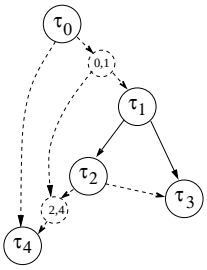


Figure 4: Mapped and scheduled task graph with pseudo communication tasks and edges

$$i(\epsilon_n) \cap i(\epsilon_m) = \varnothing, \forall(\epsilon_n, \epsilon_m) \tag{8}$$

Note that the power dissipation $P_{max}(\tau)$ of each task given in Equation (4) is *not* constant, but might vary from task to task, unlike in previous approaches. A constructive heuristic to solve this optimisation problem is presented in Section 4.

4. PROPOSED POWER VARIATION (PV) DVS ALGORITHM

Allowing the PE power variations among tasks makes it difficult to identify suitable scaling voltages for the tasks such that the energy dissipation is minimised. Recently, a hybrid global/local search strategy [1] has been proposed to find optimised execution voltages for each task. While this formulation of the problem might be applied to consider power variations among tasks, its implementation using a genetic algorithm combined with a local search will lead to prohibitively large computational time (with reported times of 20 minutes of 28 node task graphs) hindering a efficient exploration of the system level design space, when used in the inner most loop of the co-synthesis. The aim of this section is to introduce a new voltage scaling algorithm capable of identifying refined scaling voltages by considering the individual power dissipation based on the executed task. The presented algorithm is sufficiently fast enough to permit its use during the design space exploration phase to allow the optimisation of the scheduling and mapping towards DVS usability. The proposed scaling algorithm has a polynomial time complexity of $\mathcal{O}(m \cdot n(3n + e + n \log n))$, with $n = |\mathcal{T}|$ and $e = |\mathcal{C}|$. The factor m depends on the Δt_{min} and is introduced to capture the fact that tasks mind be extended more than once. However, this time complexity is a worst case estimation, which is unlikely to be reached due to the progressively elimination of extendable tasks during the optimisation run.

Fig. 5 gives the pseudocode of the proposed voltage scaling algorithm. Its input consists of the specification task graph G_S, a mapping of tasks to the architecture, a schedule of tasks and communications, architectural information (e.g. execution time and power dissipation of tasks), and the minimum extention time Δt_{min} (defined to prevent insignificant small extentions, which would slow down the DVS algorithm). Initially (line 01), the specification task graph is transformed into a *mapped and scheduled task graph* (MSTG) to capture the mapping and scheduling information into the task graph. This is carried out in two steps. First, communications between different PEs are converted into pseudo

Algorithm: *PV_DVS_OPTIMISATION*

Input: - task graph $G_S(\mathcal{T}, \mathcal{C})$
- mapping and schedule
- architectural information
- minimum extention time Δt_{min}

Output: - energy optimised voltages $V_{dd}(\tau)$
- dissipated dynamic energy E

```
01: G_S ↷ MSTG
02: Q_E ← ∅
03: for all (τ ∈ T_d)  {Δt_d(τ) := θ(τ) − t_E(τ)}
04: for all (τ ∈ T) {calculate t_ε}
05: for all (τ ∈ T) {if t_ε ≥ Δt_min then Q_E := Q_E + τ}
06: Δt = min t_ε / |Q_E|, if Δt < Δt_min then Δt = Δt_min
07: for all (τ ∈ Q_E) {calculate ΔE(τ)}
08: reorder Q_E in decreasing order of ΔE
09: while (Q_E ≠ ∅) {
10:       select task τ_ΔEmax ∈ Q_E
11:       t(τ_ΔEmax) := t(τ_ΔEmax) + Δt
12:       update E_τΔEmax
13:       for all (τ ∈ T) {update t_S, t_E and t_ε}
14:       for all (τ ∈ Q_E)
                {if (t_ε(τ) < Δt_min) ∨ (V_dd(τ) ≤ V_t(τ))
                    then Q_E := Q_E − τ}
15:       Δt = min t_ε / |Q_E|, if Δt < Δt_min then Δt = Δt_min
16:       for all (τ ∈ Q_E) {update ΔE(τ)}
17:       reorder Q_E in decreasing order of ΔE
18: }
19: for all (τ ∈ T) {return V_dd(τ)}
20: return E_Σ
21: MSTG ↷ G_S
```

Figure 5: Pseudocode of the proposed variable power DVS algorithm (PV-DVS)

communication tasks to allow an unification of communications and tasks. Second, for each successive execution of tasks on same resources a pseudo edge is introduced between the these two tasks [4], if it does not already exists. In this way it becomes possible to easily and fast traverse the task graph in the chronological correct order by using a breadth first search algorithm (linear time complexity). An example MSTG is shown in Fig. 4, which is the transformed version of the task graph given in Fig. 1, for the schedule of Fig. 2. The next step (line 02) initialises the extentable task queue $\mathbb{Q}_E$. This is followed (line 03) by the computation of the deadline slacks Δt_d of all deadline tasks $\tau \in \mathcal{T}_d$. Having determined these slacks the algorithm can now proceed (line 04) to identify extendable tasks and calculate their maximal expendability t_ϵ. This is done in a reversed breadth-first search manner. All task extendable by at least Δt_{min} are added to the queue of extendable tasks $\mathbb{Q}_E$ (line 05). Next (line 06), the extention time Δt is computed as the minimal extention time found in $\mathbb{Q}_E$, divided by the number of elements in $\mathbb{Q}_E$. If Δt is smaller then Δt_{min} it is set to Δt_{min} to avoid unnecessary optimisation runs, which would increase m in the complexity function $\mathcal{O}$. In the last two steps of the initialisation phase (line 07–08) the energy gradients ΔE for all extendable tasks are calculated and the extendable task queue $\mathbb{Q}_E$ is reordered in decreasing order of the energy gradients. The energy gradient value reflects the energy gain if a particular task is extended by Δt. Therefore it is this energy gradient that allows the algorithm to identify the

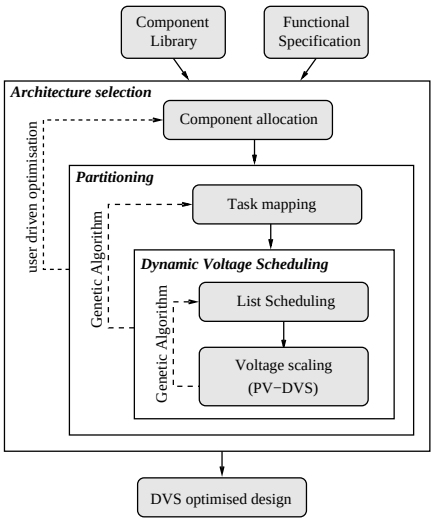


Figure 6: Typical system level co-synthesis including DVS

tasks which will result in the highest energy reduction and therefore an improved voltage selection is possible. The following loop (line 09) is repeated until no extendable tasks are left. In each iteration a single task with the highest energy reduction gradient ΔE (the first task in the queue) is picked out of the the extendable task queue (line 10). This task is then (line 11) extended by Δt and its energy value is updated (line 12). Since the extention of the task might influence successor tasks, the extention needs to be propagated through the task graph (line 13). This is done by traversing all nodes and edges of the MSTG from the extended task to the end tasks, using a breadth-first search algorithm. In the next step (line 14) inextensible tasks are removed from the extendable task queue $\mathbb{Q}_E$ when either, their extendabilty t_ϵ is smaller then Δt_{min}, or now further voltage reduction is possible due to the limited supply voltage range. After this (line 15), the extention time Δt is recalculated and all energy gradients of extendable task are recomputed (line 16). Since the energy gradients might have change the queue $\mathbb{Q}_E$ is reordered (line 17). If no extendable tasks are left the algorithm finishes by returning the scaled voltages (line 19) and the total dissipated energy (line 20). Furthermore, the MSTG is transformed back into a "normal" TG, in order to allow for the next scheduling optimisation step, which leads to a different assignment of pseudo edges.

5. EXPERIMENTAL RESULTS

This section demonstrates through the use of benchmark examples that the consideration of PE power variations leads to energy savings in distributed embedded systems. The design flow outlined in Fig. 6, along with the proposed DVS algorithm, have been implemented using C++ on a Pentium-III/750 PC with 128MB RAM running the Linux operating system. Scheduling and mapping are optimised using techniques based on genetic algorithms similar to the reported approaches in [7] and [8]. However, unlike [8] we do not applied a hole filling technique in the list scheduler, which tries to fill idle times with suitable task, since such idle times are exploited by the DVS technique. Due to the space restriction we can not give a detailed elucidation here. The six benchmarks we use here are partly (*hou*) from literature [12] and partly (*tgff*) generated by TGFF [6]. The allocated components for the *hou* examples are the optimised architectures given in [12] (DVS extended), while for the *tgff* graphs an architecture was chosen randomly, with 2 to 4 PEs, such that feasible schedules and mappings could be generated. All results given here represent average values obtained by 10 optimisation runs.

Table 2 gives the results of three experiments using the above mentioned benchmarks of various complexity (i.e. number of tasks and edges). The first experiment (I) shows a comparison between a DVS approach, which neglectes the power variations (EVEN; similar to [16]) and the proposed approach (PV), which takes the power profile into account during the voltage selection. Both, techniques are applied to identical mappings of the tasks and the schedules are generated by a mobility based list scheduling algorithm. The heading E_{EV}/E_{max} refers to the ratio of the reduced energy dissipation (neglecting power variations) E_{EV} and the energy dissipation at nominal supply voltage E_{max}. It is seen that for all benchmarks the energy dissipation could be reduced. For example, in the case of *tgff1* the energy dissipation is reduced to 53.8%, compared to the system running at nominal voltage. Similar, the heading E_{PV}/E_{max} shows the ratio between the reduced energy dissipation E_{PV} using the proposed DVS algorithm and the energy dissipation at nominal supply voltage E_{max}, reflecting the energy saving achieved by the proposed approach. It can be observed that in all cases the introduced technique was able to further reduce the power consumption of the system. For benchmark *tgff1* the energy dissipation could be reduced to 29.3% of the initial power consumption. This is a further reduction of 24.5%, when compared to the DVS technique which neglects the power profile. Although the introduced DVS technique shows a computational overhead due to its iterative extention of tasks the optimisation was carried out in less $0.2s$ for all examples.

While the employment of the proposed DVS heuristic reduces energy when compared to power neglecting DVS approaches, further savings can be achieved when the heuristic is combined with a scheduling optimisation. This is shown in column of Table 2. This column gives a comparison between the energy dissipations of a GA based scheduling (GS) E_{GS} and a mobility based scheduling (MS) E_{MS}, both using the proposed PV-DVS algorithm. It can be observed that the GA based list scheduling was able to improve the energy dissipation in 5 out of 6 cases when compared to mobility based list scheduling, with reductions of up to 35.7% (*tgff3* on top of the results given in the first experiment (I) of Table 2. The computational times for this experiments varied between $0.14s$ and $32.61s$, for the GA based list scheduling.

The last experiment (III) is concered with the optimised mapping of tasks to PEs such that DVS can be further exploited and hence the energy dissipation is further reduced. Table 2 shows the results of this experiment in the Column Mapping, which gives the ratio between the mapping and scheduling optimised energy dissipation E_{MGSPV} based on the proposed PV-DVS algorithm, and the energy dissipation E_{MGSmax} of the mapping and scheduling optimised implementations running at nominal supply voltage. Overall energy reductions of up to 80.7% (*tgff4*) are achieved. Certainly, the improved results require longer computional times, which are located between $2s$ and $3070s$, compared to 1.58 and $77.24s$ when no DVS is considered.

Benchmark Example	*No. of tasks/edges*	DVS (I)		Scheduling (II)	Mapping (III)
		EVEN E_{EV}/E_{max}	PV E_{PV}/E_{max}	E_{GS}/E_{MS}	E_{MGS}/E_{MGSmax}
tgff1	8/9	0.538	0.293	0.990	0.279
tgff2	26/43	0.962	0.859	0.942	0.718
tgff3	40/77	0.772	0.448	0.643	0.452
tgff4	20/33	0.950	0.871	0.999	0.193
hou	20/29	0.927	0.781	0.776	0.585
hou_clust	8/7	0.794	0.714	1.000	0.622

Table 2: Experimental results of various benchmarks used for DVS, scheduling, and mapping optimisation

6. CONCLUSION

This paper has addressed the problem of energy minimisation in distributed embedded systems including DVS processing elements. It has been shown that it is unnecessary to assume that PE power is constant during the execution of task. Indeed, by considering the power variations of the PEs during the synthesis further energy savings can be achieved. A new heuristic taking into account the PE power variations based on the executed task has been proposed and its capability to identify suitable scaling voltages has been shown. Experimental results demonstrate that combining the proposed scaling heuristic with a genetic algorithm based design space exploration yields to substantially reduced (up to 80.7%) energy dissipations. The energy savings have been achieved in moderate computational time due to the polynominal time complexity of the proposed scaling algorithm.

Acknowledgements

The authors gratefully acknowledge Petru Eles for useful comments and suggestions on this paper. They also wish to acknowledge EPSRC for funding this research project.

7. REFERENCES

[1] N. Bambha, S. Bhattacharyya, J. Teich, and E. Zitzler. Hybrid Global/Local Search Strategies for Dynamic Voltage Scaling in Embedded Multiprocessors. In *Proc. CODES*, pages 243–248, April 2001.

[2] C. Brandolese, W. Fornaciari, F. Salice, and D. Sciuto. Energy Estimation for 32 bit Microprocessors. In *Proc. CODES*, pages 24–28, May 2000.

[3] T. D. Burd, T. A. Pering, A. J. Stratakos, and R. W. Brodersen. A Dynamic Voltage Scaled Microprocessor System. *IEEE Journal of Solid-State Circuits*, 35(11):1571–1580, November 2000.

[4] P. Chretienne, E. G. Coffman, J. K. Lenstra, and Z. Liu. *Scheduling Theory and its Applications*. John Wiley & Sons, 1995.

[5] S. Devadas and S. Malik. A Survey of Optimization Techniques Targeting Low Power VLSI Circuits. In *Proc. DAC*, pages 242–247, 1995.

[6] R. Dick, D. Rhodes, and W. Wolf. TGFF: Task Graphs for free. In *Proc. CODES*, pages 97–101, March 1998.

[7] R. P. Dick and N. K. Jha. MOGAC: A Multiobjective Genetic Algorithm for Hardware-Software Co-Synthesis of Distributed Embedded Systems. *IEEE Trans. on CAD*, 17(10):920–935, Oct 1998.

[8] M. Grajcar. Genetic List Scheduling Algorithm for Scheduling and Allocation on a Loosely Coupled Heterogeneous Multiprocessor System. In *Proc. DAC*, pages 280–285, 1999.

[9] F. Gruian and K. Kuchcinski. LEnS: Task Scheduling for Low-Energy Systems Using Variable Supply Voltage Processors. In *Proc. ASP-DAC*, pages 449–455, Jan 2001.

[10] V. Gutnik and A. Chandrakasan. Embedded Power Supply for Low-Power DSP. *IEEE Trans. on VLSI Systems*, 5(4), 425–435 1997.

[11] I. Hong, D. Kirovski, G. Qu, M. Potkonjak, and M. B. Srivastava. Power Optimization of Variable-Voltage Core-Based Systems. *IEEE Trans. on Computer-Aided Design*, 18(12):1702–1714, Dec 1999.

[12] J. Hou and W. Wolf. Process Partitioning for Distributed Embedded Systems. In *Proc. CODES*, pages 70 – 76, March 1996.

[13] T. Ishihara and H. Yasuura. Voltage Scheduling Problem for Dynamically Variable Voltage Processors. In *Proc. ISLPED*, pages 197–202, 1998.

[14] A. Klaiber. The Technology behind Crusoe Processors, January 2000. http://www.transmeta.com.

[15] Y.-T. S. Li, S. Malik, and A. Wolfe. Performance Estimation of Embedded Software with Instruction Cache Modeling. In *Proc. ICCAD*, pages 380–387, Nov. 1995.

[16] J. Luo and N. K. Jha. Power-conscious Joint Scheduling of Periodic Task Graphs and Aperiodic Tasks in Distributed Real-time Embedded Systems. In *Proc. ICCAD*, pages 357–364, Nov 2000.

[17] A. Manzak and C. Chakrabarti. Variable Voltage Task Scheduling for Minimizing Energy or Minimizing Power. In *Proc. Internation Conference on Acoustics, Speech, and Signal Processing*, pages 3239–3242, 2000.

[18] T. Okuma, T. Ishihara, and H. Yasuura. Real-Time Task Scheduling for a Variable Voltage Processor. In *Proc. ISSS*, pages 24–29, 1999.

[19] M. Weiser, B. Welch, A. Demers, and S. Shenker. Scheduling for reduced CPU energy. In *Proc. USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 13–23, 1994.

[20] W. H. Wolf. Hardware/Software Co-Design of Embedded Systems. In *Proceedings of the IEEE*, pages 967–989, July 1994.

[21] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced CPU energy. In *IEEE Symposium on Foundations of Comp. Science*, pages 374–382, 1995.