DATE04 SPECIAL SECTION

Overhead-conscious voltage selection for dynamic and leakage energy reduction of time-constrained systems

A. Andrei, M. Schmitz, P. Eles, Z. Peng and B.M. Al-Hashimi

Abstract: Dynamic voltage scaling and adaptive body biasing have been shown to reduce dynamic and leakage power consumption effectively. The authors report an optimal solution to the combined supply voltage and body bias selection problem for multiprocessor systems with imposed time constraints, explicitly taking into account the transition overheads implied by changing voltage levels, and considering both energy and time overheads. They investigate continuous voltage scaling as well as its discrete counterpart, and strongly prove NP-hardness in the discrete case. Furthermore, the continuous voltage scaling problem is formulated and solved using nonlinear programming with polynomial time complexity, while for the discrete problem they use mixed integer linear programming. Extensive experiments, conducted on several benchmarks and a real-life example, are used to validate the approaches.

1 Introduction

Embedded computing systems in portable devices need to be energy efficient, yet they have to deliver adequate performance to the often computationally expensive applications, such as voice processing and multimedia. Owing to the diversity of the applications that run within a single device and their different degrees of parallelism, the workload imposed on the embedded system is nonuniform over time. This introduces slack times during which the system can reduce its performance to save energy. Two system-level approaches that allow an energy/performance tradeoff during run-time of the application are dynamic voltage scaling (DVS) [1–3] and adaptive body biasing (ABB) [2, 4]. While DVS aims to reduce the dynamic power consumption by scaling down operational frequency and circuit supply voltage V_{dd}, ABB is effective in reducing the leakage power by scaling down frequency and increasing the threshold voltage V_{th} through body biasing. To date, most research efforts at the system level have been devoted to DVS, since the dynamic power component *had been* dominating. Nonetheless, the trend in deep-submicron CMOS technology to reduce the supply voltage levels and consequently the threshold voltages (to maintain peak performance) is resulting in the fact that a substantial portion of the overall power dissipation will be due to leakage currents [4, 5]. This makes the adaptive body-biasing approach and its combination with dynamic voltage scaling indispensable for energy-efficient designs in the foreseeable future.

IEE Proceedings online no. 20045055

doi: 10.1049/ip-cdt:20045055

Paper first received 29th June and in revised form 14th October 2004

A. Andrei, M. Schmitz, P. Eles and Z. Peng are with the Department of Computer and Information Science, Linköping University, SE-58 183 Linköping, Sweden

B.M. Al-Hashimi is with the Department of Electronics and Computer Science, University of Southampton, Southampton, SO17 1BJ, UK

Voltage selection approaches can be broadly classified into on-line and off-line techniques. In the following, we restrict ourselves to the off-line techniques since the presented approaches fall into this category, where the scaled supply voltages are calculated before design time and then applied at run-time according to the precalculated voltage schedule.

There has been a considerable amount of work on dynamic voltage scaling. Yao *et al.* [3] proposed the first DVS approach for single processor systems which can dynamically change the supply voltage over a continuous range. Ishihara and Yasuura [1] modelled the discrete voltage selection problem using an integer linear programming (ILP) formulation. Kwon and Kim [6] proposed a linear programming (LP) solution for the discrete voltage selection problem with uniform and nonuniform switched capacitance. Although this gives the impression that this problem can be solved optimally in polynomial time, we will show in this paper that the discrete voltage selection problem is indeed strongly NP-hard and, hence, no optimal solution can be found in polynomial time, for example using LP. Dynamic voltage scaling has also been successfully applied to heterogeneous distributed systems, in which numerous processing elements interact via a communication infrastructure, mostly using heuristics [7–9]. Zhang *et al.* [10] approached continuous supply voltage selection in distributed systems using an ILP formulation. They solved the discrete version of the problem through an approximation.

While the approaches mentioned above scale the supply voltage V_{dd} only and neglect leakage power consumption, Kim and Roy [4] proposed an adaptive body-biasing approach, in their work referred to as dynamic V_{th} scaling, for active leakage power reduction. They demonstrate that the efficiency of ABB will become, with advancing CMOS technology, comparable to DVS. Duarte *et al.* [11] analyse the effectiveness of supply and threshold voltage selection, and show that simultaneous adjustment of both voltages provides the highest savings. Martin *et al.* [2] presented an approach for combined dynamic voltage scaling and adaptive body biasing. At this point we should emphasise that, as opposed to these three approaches, we investigate in

this paper how to select voltages for a set of tasks, possibly with dependencies, which are executed on multiprocessor systems under real-time constraints. Furthermore, as opposed to our work, the techniques mentioned above *neglect* the energy and time overheads imposed by voltage transitions. Notable exceptions are [12–14], yet their algorithms ignore leakage power dissipation and body biasing, and further they do not guarantee optimality. In this work, we consider simultaneous supply voltage selection and body biasing, in order to minimise dynamic as well as leakage energy. In particular, we investigate four different notions of the combined dynamic voltage scaling and adaptive body-biasing problem — considering continuous and discrete voltage selection with and without transition overheads. A similar problem for continuous voltage selection has been recently formulated in [15]. However, it is solved using a suboptimal heuristic. The presented work makes the following contributions:

(*a*) We consider both supply voltage and body-bias voltage selection at the system level, where several tasks with dependencies execute a time-constrained application on a multiprocessor system.
(*b*) Four different voltage selection schemes are formulated as nonlinear programming (NLP) and mixed integer linear programming (MILP) problems which can be solved optimally. The formulations are equally applicable to single- and multi-processor systems.
(*c*) We prove that discrete voltage selection with and without the consideration of transition overheads in terms of energy and time is strongly NP-hard, while the continuous voltage selection cases can be solved in polynomial time (with an arbitrary given approximation $\varepsilon > 0$).

To the best of our belief, we report novel optimal voltage scheduling techniques that account for transition overheads in terms of energy and delay.

2 Preliminaries

2.1 Architectural model and system specification

In this paper we consider embedded systems which are realised as heterogeneous distributed architectures. Such architectures consist of several different processing elements (PEs), such as programmable microprocessors, ASIPs, FPGAs and ASICs, some of which feature DVS and ABB capability These computational components communicate via an infrastructure of communication links (CLs), like buses and point-to-point connections. We define $\mathcal{P}$ and $\mathcal{L}$ to be the sets of all processing elements and all links, respectively. An example architecture is shown in Fig. 1*a*. The functionality of data-flow intensive applications, such as voice processing and multimedia, can be captured by task graphs $G_S(\mathcal{T}, C)$. Nodes $\tau \in \mathcal{T}$ in these directed acyclic graphs represent computational tasks, while edges $\gamma \in \mathcal{C}$ indicate data dependencies between these tasks (communications). Tasks require a finite number of clock cycles *NC* to be executed, depending on the PE to which they are mapped. Further, tasks are annotated with deadlines *dl* that have to be met during application run-time. If two dependent tasks are assigned to different PEs, π_x and π_y with $x \neq y$, then the communication takes place over a CL, involving a certain amount of communication time and power.

We assume that the task graph is mapped and scheduled onto the target architecture, i.e. it is known where and in which order tasks and communications take place. Figure 1*a* shows an example task graph that has been mapped onto an architecture and Fig. 1*b* depicts a possible execution order. On top of the precedence relations given by data dependencies between tasks, we introduce additional precedence relations $r \in \mathcal{R}$, generated as result of scheduling tasks mapped to the same PE and communications mapped on the same CL. In Fig. 1*c* the dependencies $\mathcal{R}$ are represented as dotted edges. We define the set of all edges as $\mathcal{E} = \mathcal{C} \cup \mathcal{R}$. Further, we define the set $\mathcal{E}^{\bullet} \subseteq \mathcal{E}$ of edges, as follows: an edge $(i, j) \in \mathcal{E}^{\bullet}$ if it connects task τ_i with its immediate successor τ_j (according to the schedule), where τ_i and τ_j are mapped on the same PE.

2.2 Power and delay models

Digital CMOS circuitry has two major sources of power dissipation: (*a*) dynamic power P_{dyn}, which is dissipated whenever active computations are carried out (switching of logic states), and (*b*) leakage power P_{leak} which is consumed whenever the circuit is powered, even if no computations are performed. The dynamic power is expressed by [2, 16]

$$P_{dyn} = C_{eff} f V_{dd}^2 \tag{1}$$

where C_{eff}, f and V_{dd} denote the effective charged capacitance, operational frequency and circuit supply voltage, respectively. Although, until recently, the dynamic power dissipation had been dominating, the trend to reduce the overall circuit supply voltage and consequently threshold voltage is raising concerns about the leakage currents – for near future technology (<70 nm) it is

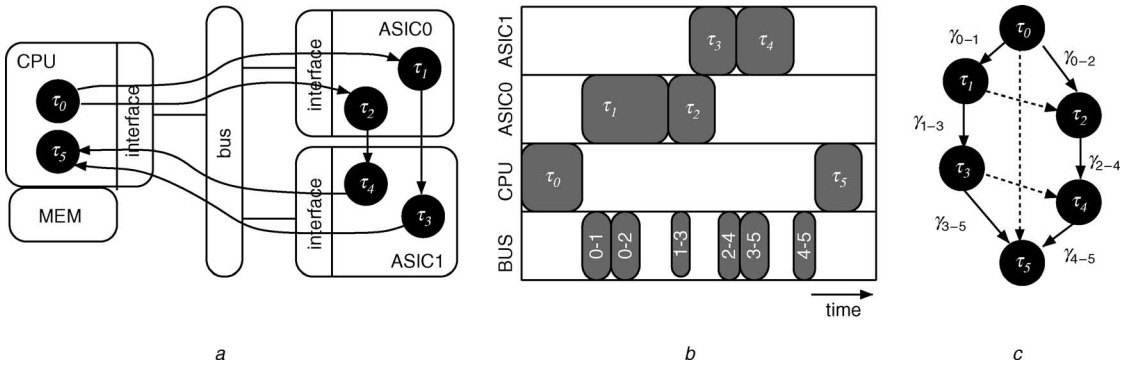

Fig. 1 *System models*
a Target architecture including mapped task graph
b Scheduled tasks and communications
c Scheduled task graph

expected that leakage will account for more than 50% of the total power. The leakage power is given by [2]

$$P_{leak} = L_g V_{dd} K_3 e^{K_4 V_{dd}} e^{K_5 V_{bs}} + |V_{bs}| I_{Ju} \quad (2)$$

where V_{bs} is the body-bias voltage and I_{Ju} represents the body junction leakage current (constant for a given technology). The fitting parameters K_3, K_4 and K_5 denote circuit technology dependent constants and L_g reflects the number of gates. For clarity reasons we maintain the same indices as used in [2], where also actual values for these constants are given. Please note that the leakage power is more strongly influenced by V_{bs} than by V_{dd}, due to the fact that the constant K_5 (e.g. Crusoe $K_5 = 4.19$) is larger than the constant K_4 (e.g. Crusoe $K_4 = 1.83$).

Nevertheless, scaling the supply and the body-bias voltage, to reduce the power consumption, has a side-effect on the circuit delay d and hence the operational frequency [2, 16],

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

where α reflects the velocity saturation imposed by the technology used (common values $1.4 \leq \alpha \leq 2$), L_d is the logic depth, and K_1, K_2, K_6 and V_{th1} are circuit dependent constants.

Another important issue, which is often overlooked in voltage scaling approaches, is the consideration of transition overheads, i.e. each time the processor's supply voltage and body-bias voltage are altered, the change requires a certain amount of extra energy and time. These energy $\varepsilon_{k,j}$ and delay $\delta_{k,j}$ overheads, when switching from V_{dd_k} to V_{dd_j} and from V_{bs_k} to V_{bs_j}, are given by [2]:

$$\varepsilon_{k,j} = C_r |V_{dd_k} - V_{dd_j}|^2 + C_s |V_{bs_k} - V_{bs_j}|^2 \quad (4)$$

$$\varepsilon_{k,j} = \max(p_{Vdd} |V_{dd_k} - V_{dd_j}|, p_{Vbs} |V_{bs_k} - V_{bs_j}|) \quad (5)$$

where C_r denotes power rail capacitance, and C_s the total substrate and well capacitance. Since transition times for V_{dd} and V_{bs} are different, the two constants p_{Vdd} and p_{Vbs} are used to calculate both time overheads independently. Considering that supply and body-bias voltage can be scaled in parallel, the transition overhead $\delta_{k,j}$ depends on the maximum time required to reach the new voltage levels.

Voltage scaling is only rewarding if the energy saved through optimised voltages is not outdone by the transition overheads in energy. Furthermore, it is obvious that disregarding transition time overhead can seriously affect the schedulablity of real time systems.

2.3 Mathematical programming

In this Section we briefly outline some useful mathematical programming issues, which are relevant for the rest of the paper. Mathematical programming offers methods for solving problems of minimising or maximising an objective function $f(x_1, \ldots, x_n)$, with respect to a set of m constraints $g_j(x_1, \ldots, x_n) \leq c_j \, (j = 1, \ldots, m)$ and bounds for the n variables $(lb_i \leq x_i \leq ub_i, i = 1, \ldots, n)$. If both the objective function f and the constraints g_j are linear functions, the problem is called linear programming (LP). Further, if some of the variables are restricted to the integer domain, the problem is called mixed integer linear programming (MILP). If either f or g_j are nonlinear functions, we have a nonlinear programming (NLP) problem. If both f and g_j are convex functions and the variables range over a continuous domain, the problem is called convex nonlinear programming (convex NLP). Solving MILP problems was proved to be NP-complete [Note 1]. For LP problems, there exist polynomial algorithms. For convex NLP, efficient algorithms are available [17] that solve the problem within a given arbitrary small $\varepsilon > 0$ approximation error in polynomial time.

3 Motivational example

To demonstrate the influence of the transition overheads in terms of energy and delay, consider the following motivational example. For clarity reasons we restrict ourselves here to a single processor system that offers three voltage modes, $m_1 = (1.8\,\text{V}, -0.3\,\text{V})$, $m_2 = (1.5\,\text{V}, -0.45\,\text{V})$ and $m_3 = (1.2\,\text{V}, -0.8\,\text{V})$, where $m_z = (V_{dd_z}, V_{bs_z})$. The rail and substrate capacitance are given as $C_r = 10\,\mu\text{F}$ and $C_s = 40\,\mu\text{F}$. The processor needs to execute two consecutive tasks (τ_1 and τ_2) with a deadline of 0.225 ms. Figure 2*a* shows a possible voltage schedule. As we can observe, each of the two tasks is executed in two different modes: task τ_1 executes first in mode m_2 and then in mode m_1, while task τ_2 is initially executed in mode m_3 and then in mode m_2. The total energy consumption of this schedule is the sum of the energy dissipation in each mode $E = 9 + 15 + 4.5 + 7.5 = 36\,\mu\text{J}$. However, if this voltage schedule is applied to a *real* voltage-scalable processor, the resulting schedule will be influenced by transition overheads, as shown in Fig. 2*b*. Here the processor requires a finite time to adapt to the new execution mode. During this adaption no computations can be performed [18, 19], i.e. the task execution is delayed, which, in turn, increases the schedule length such that the imposed deadline is violated. Moreover, transitions not only require time, they also cause an additional energy dissipation. For instance, in the given schedule, the first transition overhead O_1 from mode m_2 and m_1 requires an energy of $10\,\mu\text{F}(1.8\,\text{V} - 1.5\,\text{V})^2 + 40\,\mu\text{F}(0.3\,\text{V} - 0.45\,\text{V})^2 = 1.8\,\mu\text{J}$, based on (4). Similarly, the energy overheads for transitions O_2 and O_3 can be calculated as $13.6\,\mu\text{J}$ and $5.8\,\mu\text{J}$, respectively. The overall energy dissipation of the realistic schedule shown in Fig. 2*b* accumulates to $36 + 1.8 + 13.6 + 5.8 = 57.2\,\mu\text{J}$.

Let us consider a second possibility of ordering the modes, as given in Fig. 2*c*. Compared to the schedule in Fig. 2*a*, the mode activation order in Fig. 2*c* has been swapped for both tasks. As long as the transition overheads are neglected, the energy consumption of the two schedules is identical. However, applying the second activation order to a real processor would result in the schedule shown in Fig. 2*d*. We can observe that this schedule exhibits only two mode transitions (O_1 and O_3) within the tasks (intra-switches), while the switch between the two tasks O_2 (inter-switch) has been eliminated. The overall energy consumption has been reduced to $E = 43.6\,\mu\text{J}$, a reduction by 23.8% compared to the schedule given in Fig. 2*b*. Further, the elimination of transition O_2 reduces the overall schedule length, such that the imposed deadline is satisfied. With this motivational example we have demonstrated the effects that transition overheads can have on the energy consumption and the timing behaviour and the impact of taking them into consideration when elaborating the voltage schedule.

Note 1: For some subclasses, e.g. convex objectives with linear constraints, there exist polynomial algorithms that solve the MILP formulation.

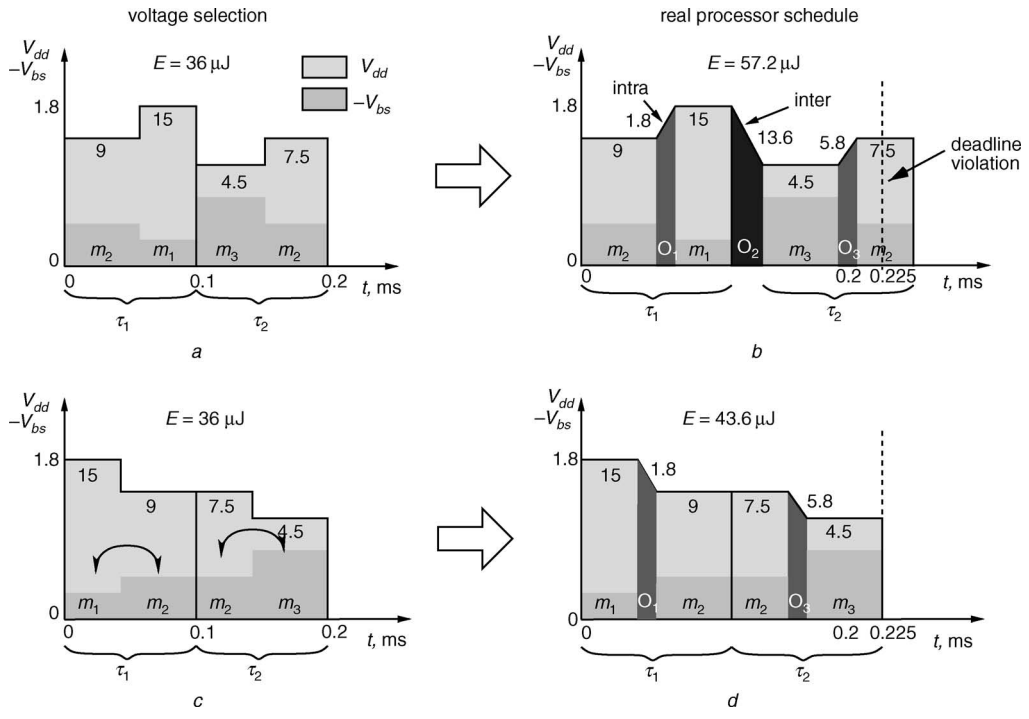

Fig. 2 *Influence of transition overheads*

a Before reordering, without overheads
b Before reordering, with overheads
c After reordering, without overheads
d After reordering, with overheads

However, the approaches presented in this paper not only achieve energy efficiency by considering transition overheads, but further take into account the simultaneous scaling of the supply voltage V_{dd} and body-bias voltage V_{bs}. To illustrate the advantage of this simultaneous scaling over supply voltage scaling only, consider the following example.

Figure 3 shows two optimal voltage schedules for a set of three tasks (τ_1, τ_2 and τ_3), executing in two possible voltage modes. While the first schedule relies on V_{dd} scaling only (i.e. V_{bs} is kept constant), the second schedule corresponds to the simultaneous scaling of V_{dd} and V_{bs}. Please note that the Figures depict the dynamic and the leakage power dissipation as a function of time, unlike Fig. 2, which showed V_{dd} and V_{bs} as a function of time. For simplicity we neglect transition overheads in this example. Further, we consider processor parameters that correspond to CMOS technology (<70 nm), which leads to a leakage power consumption close to 40% of the total power consumed (at the mode with the highest performance).

Let us consider the first schedule in which the tasks are executed either at $V_{dd1} = 1.8$ V, or $V_{dd2} = 1.5$ V, while V_{bs1} and V_{bs2} are kept at 0 V. In accordance, the system dissipates $P_{dyn1} = 100$ mW and $P_{leak1} = 75$ mW in mode 1 running at 700 MHz, while $P_{dyn2} = 49$ mW and $P_{leak2} = 45$ mW in mode 2 running at 525 MHz, as observable from the Figure. We have also indicated the individual energy consumed in each of the active modes, separating between dynamic and leakage energy. Correspondingly, the total leakage and the total dynamic energies of the schedule in Fig. 3*a* are given by 13.56 µJ and 16.17 µJ, respectively. This results in a total energy consumption of 29.73 µJ.

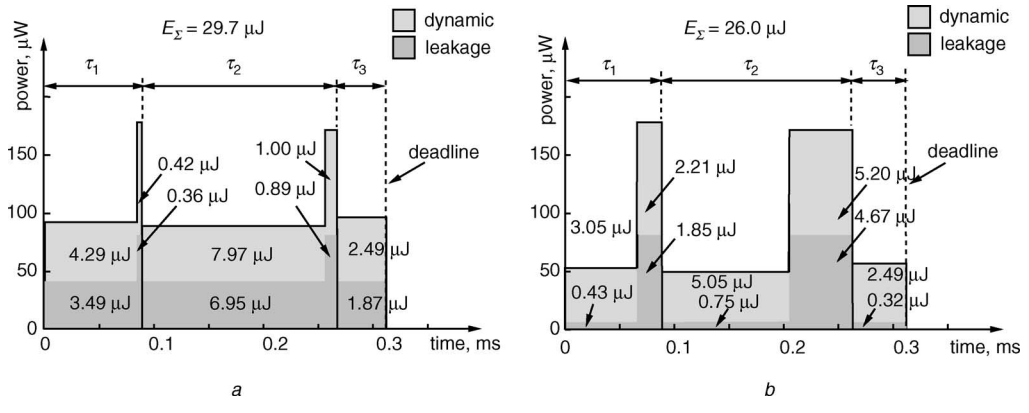

Fig. 3 *Influence of V_{bs} scaling*

a V_{dd} scaling only
b Simultaneous V_{dd} and V_{bs} scaling

Consider now the schedule given in Fig. 3*b*, where tasks are executed at two different voltage settings for V_{dd} and V_{bs} ($m_1 = (1.8\,V, 0V)$ and $m_2 = (1.5\,V, -0.4\,V)$). Since the voltage settings for mode m_1 did not change, the system runs at 700 MHz and dissipates $P_{dyn1} = 100\,\text{mW}$ and $P_{leak1} = 75\,\text{mW}$. In mode m_2 the system performance with 480 Mhz and dissipates $P_{dyn2} = 49\,\text{mW}$ and $P_{leak2} = 5\,\text{mW}$. There are two main differences to observe compared to the schedule in Fig. 3*a*. Firstly, the leakage power consumption during mode m_2 is considerably smaller than the leakage power given in the schedule of Fig. 3*a*; this is due to the fact that m_2 reduces the leakage through a body-bias voltage of $-0.4\,\text{V}$ (see (2)). Secondly, the high voltage mode m_1 is active for more time; which can be explained by the fact that scaling V_{bs} during mode m_2 requires the reduction of the operational frequency (see (3)). Hence, to meet the system deadline, high performance mode m_1 has to compensate for this delay. Nevertheless, the total leakage and dynamic energies result in 8.02 µJ and 18.00 µJ, respectively. Although here the dynamic energy was increased from 16.17 µJ to 18.0 µJ, compared to the first schedule, the leakage was reduced from 13.56 µJ to 8.02 µJ. The overall energy dissipation becomes then 26.02 µJ, a reduction by 12.5%. This small illustrative examples shows the advantage of simultaneous V_{dd} and V_{bs} scaling compared to V_{dd} scaling only.

4 Problem formulation

Consider a set of tasks with precedence constraints $\mathcal{T} = \{\tau_i\}$ that have been mapped and scheduled on a set of variable voltage processors. For each task τ_i its deadline dl_i, its number of clock cycles to be executed NC_i and the switched capacitance C_{eff_i} are given. Each processor can vary its supply voltage V_{dd} *and* body bias voltage V_{bs} within certain continuous ranges (for the continuous problem), or within a set of discrete voltages pairs $m_z = \{(V_{dd_z}, V_{bs_z})\}$ (for the discrete problem). The power dissipations (leakage, dynamic) and the cycle time (processor speed) depend on the selected voltage pair (mode). Tasks are executed cycle by cycle, and each cycle can potentially execute at a different voltage pair, i.e. at a different speed. Our goal is to find voltage pair assignments for each task such that the individual task deadlines are met and the total energy consumption is minimal. Furthermore, whenever the processor has to alter the settings for V_{dd} and/or V_{bs}, a transition overhead in terms of energy and time is required (see (4) and (5)).

For reasons of clarity we introduce the following four distinctive problems which will be considered in this paper: (a) continuous voltage scaling with no consideration of transition overheads (CNOH); (b) continuous voltage scaling with consideration of transition overheads (COH); (c) discrete voltage scaling with no consideration of transition overheads (DNOH); and (d) discrete voltage scaling with consideration of transition overheads (DOH).

5 Optimal continuous voltage selection

In this Section we consider that supply and body-bias voltage of the processors in the system can be selected within a certain continuous range. We first formulate the problem neglecting the transition overheads (Section 5.1, CNOH) and then extend this formulation to include the overheads in energy and delay (Section 5.2, COH).

5.1 Continuous voltage selection without overheads (CNOH)

We can model the continuous voltage scaling problem excluding the consideration of transition overheads (the CNOH problem), using the following nonlinear problem formulation.

Minimise

$$\sum_{k=1}^{|\mathcal{T}|} \underbrace{(NC_k C_{eff_k} V_{dd_k}^2)}_{E_{dyn_k}} + \underbrace{L_g(K_3 V_{dd_k} e^{k_4 V_{dd_k}} e^{K_5 V_{bs_k}} + I_{Ju}|V_{bs_k}|)t_k)}_{E_{leak_k}} \tag{6}$$

subject to:

$$t_k = NC_k \frac{(K_6 L_d V_{dd_k})}{((1+K_1)V_{dd_k} + K_2 V_{bs_k} - V_{th_1})^{\alpha}} \tag{7}$$

$$D_k + t_k \leq D_t \quad \forall (k,l) \in \mathcal{E} \tag{8}$$

$$D_k + t_k \leq dl_k \quad \forall \tau_k \text{ that have a deadline} \tag{9}$$

$$D_k \geq 0 \tag{10}$$

$$V_{dd_{\min}} \leq V_{dd_k} \leq V_{dd_{\max}} \text{ and } V_{bs_{\min}} \leq V_{bs_k} \leq V_{bs_{\max}} \tag{11}$$

The variables that need to be optimised in this formulation are the task execution times t_k, the task start times D_k as well as the voltages V_{dd_k} and V_{bs_k}. The whole formulation can be explained as follows. The total energy consumption, which is the combination of dynamic and leakage energy, has to be minimised, as in (6) [Note 2]. The minimisation has to comply with the following relations and constraints. The task execution time has to be equivalent to the number of clock cycles of the task multiplied by the circuit delay for a particular V_{dd_k} and V_{bs_k} setting, as expressed by (7). Given the execution time of the tasks, it becomes possible to express the precedence constraints between tasks (see (8)), i.e. a task τ_l can only start its execution after all its predecessor tasks τ_k have finished their execution $(D_k + t_k)$. Predecessors of task τ_l are all tasks τ_k for which there exists an edge $(k,l) \in E$. Similarly, tasks with deadlines have to be completed $(D_k + t_k)$ before their deadlines dl_k are exceeded (see (9)). Task start times have to be positive (see (10)) and the imposed voltage ranges should be respected (see (11)). It should be noted that the objective (given in (6)) as well as the task execution time (see (7)) are convex functions. Hence, the problem falls into the class of general convex nonlinear optimisation problems. As outlined in Section 2.3, such problems can be efficiently solved in polynomial time (with a given arbitrary precision $\varepsilon > 0$). For clarity reasons, in this paper, we did not include communication issues into the constraints and objective function. Nevertheless, they can be included in a straightforward way, by modelling communication links as non-scalable processors and communications as tasks mapped to such processors.

Note 2: Please note that *abs* and *max* operations cannot be used directly in mathematical programming, yet there exist standard techniques to overcome this limitation by equivalent formulations [20].

5.2 Continuous voltage selection with overheads (COH)

In this Section we modify the previous formulation in order to take transition overheads into account (COH problem) The following formulation highlights the modifications.

Minimise

$$\underbrace{\sum_{k=1}^{|\mathcal{T}|}(E_{dyn_k}+E_{leak_k})}_{\text{task energy dissipation}}+\underbrace{\sum_{(k,j)\in\mathcal{E}^\bullet}\varepsilon_{k,j}}_{\text{transition energy overhead}} \tag{12}$$

subject to

$$D_k+t_k+\delta_{k,j}\le D_j \quad \forall(k,j)\in\mathcal{E}^\bullet \tag{13}$$

$$\delta_{k,j}=\max(p_{Vdd}|V_{dd_k}-V_{dd_j}|,\ p_{Vbs}|V_{bs_k}-V_{bs_j}|) \tag{14}$$

As we can see, the objective function (12) now additionally accounts for the transition overheads in terms of energy. The energy overheads can be calculated according to (4) for all consecutive tasks τ_k and τ_j on the same processor ($\mathcal{E}^\bullet$ is defined in Section 2.1). However, scaling voltages not only require energy but introduce delay overheads as well. These overheads might delay the start times of subsequent tasks. Therefore, we introduce an additional constraint similar to (8), which states that a task τ_j can only start after the execution of its predecessor $\tau_k(D_k+t_k)$ on the same processor and after the new voltage mode is reached $(\delta_{k,j})$. This constraint is given in (13). The delay penalties $\delta_{k,j}$ are introduced as a set of new variables and are constrained subject to (14). Similar to the CNOH formulation, the COH model is a convex nonlinear problem, i.e. it can be solved in polynomial time.

6 Optimal discrete voltage selection (DNOH)

In the preceding Section, we have shown how continuous voltage scaling can be solved optimally in polynomial time. Voltage scaling was performed for both V_{dd} and V_{bs}. Furthermore, the consideration of transition overheads was introduced into the model. These approaches provide a theoretical lower bound on the possible energy savings. In reality, however, processors are restricted to a discrete set of V_{dd} and V_{bs} voltage pairs. In this Section we investigate the discrete voltage selection problem without and with the consideration of overheads. We will also analyse the complexity of the discrete voltage selection problem.

6.1 Problem complexity

Theorem 1: The discrete voltage selection problem is NP-hard.

Proof: We prove by restriction. The discrete time–cost tradeoff (DTCT) problem is known to be NP-hard [21]. By restricting the discrete voltage selection problem (DNOH) to contain only tasks that require an execution of one clock cycle, it becomes identical to the DTCT problem. Hence, $DTCT \in DNOH$, which leads to the conclusion $DNOH \in NP$. □

The exact details of the proof are given in [20]. Note that the problem remains NP-hard, even if we restrict it to supply voltage scaling (without adaptive body-biasing) and even if transition overheads are neglected. It should be noted that this finding renders the conclusion of [6] impossible, which states that the discrete voltage scaling problem (considered in [6] without body-biasing and overheads) can be solved optimally in polynomial time [Note 3].

6.2 Discrete voltage selection without overheads (DOH)

In the following we give a mixed integer linear programming (MILP) formulation for the discrete voltage selection problem without overheads (DNOH) We consider that processors can run in different modes $m\in\mathcal{M}$. Each mode m is characterised by a voltage pair (V_{dd_m},V_{bs_m}), which determines the operational frequency f_m, the normalised dynamic power P_{dnom_m} and the leakage power dissipation P_{leak_m}. The frequency and the leakage power are given by (3) and (2), respectively. The normalised dynamic power is given by $P_{dnom_m}=f_mV_{dd_m}^2$. Accordingly, the dynamic power of a task τ_k operating in mode m is computed as $C_{eff_k}P_{dnom_m}$. Based to these definitions, the MILP problem is formulated as follows:

Minimise

$$\sum_{k=1}^{|\mathcal{T}|}\sum_{m\in\mathcal{M}}(C_{eff_k}P_{dnom_m}t_{k,m}+P_{leak_m}t_{k,m}) \tag{15}$$

subject to:

$$D_k+\sum_{m\in\mathcal{M}}t_{k,m}\le dl_k \tag{16}$$

$$D_k+\sum_{m\in\mathcal{M}}t_{k,m}\le D_l \quad \forall(k,l)\in\mathcal{E} \tag{17}$$

$$c_{k,m}=t_{k,m}f_m \quad \text{and} \quad \sum_{m\in\mathcal{M}}c_{k,m}=NC_k \quad c_{k,m}\in\mathbb{N} \tag{18}$$

$$D_k\ge 0 \quad \text{and} \quad t_{k,m}\ge 0 \tag{19}$$

The total energy consumption, expressed by (15), is given by two sums. The inner sum indicates the energy dissipated by an individual task τ_k, depending on the time $t_{k,m}$ spent in each mode m, while the outer sum adds up the energy of all tasks. Unlike the continuous voltage scaling case, we do not obtain the voltage V_{dd} and V_{bs} directly, but rather we find out how much time to spend in each of the modes. Therefore, task execution time $t_{k,m}$ and the number of clock cycles $c_{k,m}$ spent within a mode become the variables in the MILP formulation. Of course, the number of clock cycles has to be an integer and hence $c_{k,m}$ is restricted to the integer domain. We exemplify this model graphically in Figs. 4*a* and 4*b*. Figure 4*a* shows the schedule of two tasks executing each at two different voltage settings (two modes out of three possible modes). Task τ_1 executes for 20 clock cycles in mode m_2 and for 10 clock cycles in m_1, while task τ_2 runs for 5 clock cycles in m_3 and 15 clock cycles in m_2. The same is captured in Fig. 4*b* in what we call a mode model. The modes that are not active during a task's run-time have the corresponding time and number of clock cycles 0 (mode m_3 for τ_1 and m_1 for τ_2). The overall execution time of task τ_k is given as the sum of the times spent in each mode $(\Sigma_{m\in\mathcal{M}}t_{k,m})$. It should be noted that the model in Fig. 4*b* does not capture the order in which modes are activated; it solely expresses how many clock cycles

Note 3: The flaw in [6] lies in the fact that the number of clock cycles spent in a mode is not restricted to be integer.

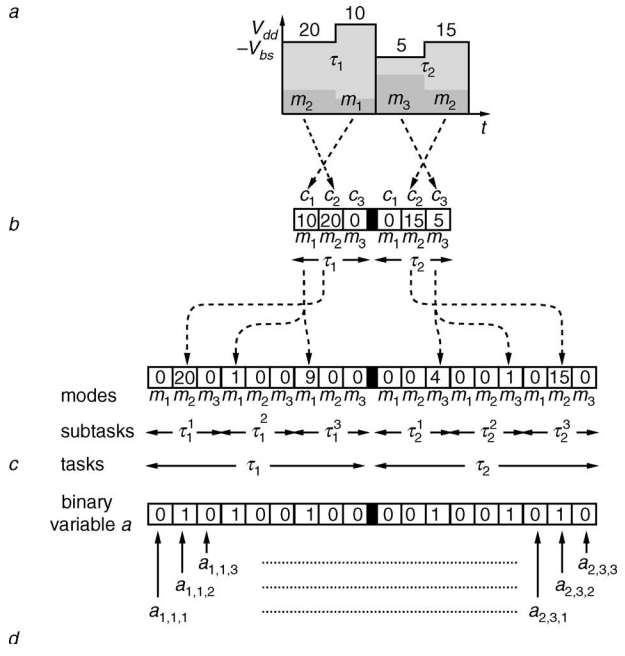

Fig. 4 *Discrete mode models*

a Schedule and mode execution order
b Tasks and clock cycles in each mode (mode execution order is not captured)
c Solution vector with division of tasks into subtasks and modes (mode execution order is captured)
d Auxiliary variables

are spent in each mode. Equation (16) ensures that all the deadlines are met and (17) maintains the correct execution order given by the precedence relations. The relation between execution time and number of clock cycles as well as the requirement to execute all clock cycles of a task are expressed in (18). Additionally, task start times D_k and task execution times have to be equal or larger than zero, as given in (19).

6.3 Discrete voltage selection with overheads

We now proceed with the incorporation of transition overheads in the MILP formulation given in Section 6.2 Obviously, the order in which the modes are activated has an influence on the transition overheads, as we have already demonstrated in Section 3. Nevertheless, the formulation in Section 6.2 omits information regarding the activation order of modes. For instance, from Fig. 4*b*, we cannot tell if for task τ_1, mode m_1 or m_2 is active first. We introduce the following extensions needed in order to take both delay and energy overheads into account. Given m operational modes, the execution of a single task τ_k can be subdivided into m subtasks $\tau_k^s, s = 1, \ldots, m$. Each subtask is executed in one and only one of the m modes. Subtasks are further subdivided into m slices, each corresponding to a mode. This results in $m \times m$ slices for each task. Figure 4*c* depicts this model, showing that task τ_1 runs first in mode m_2, then in mode m_1, and that τ_2 runs first in mode m_3, then in m_2. This ordering is captured by the subtasks: the first subtask of τ_1 executes 20 clock cycles in mode m_2, the second subtask executes one clock cycle in m_1 and the remaining 9 cycles are executed by the last subtask in mode m_1; τ_2 executes in its first subtask 4 clock cycles in mode m_3, 1 clock cycle is executed during the second subtask in mode m_3, and the last subtask executes 15 clock cycles in mode m_2. Note that there is no overhead between subsequent subtasks that run in the same mode. For instance, the two subtasks τ_1^2 1 and τ_1^3 run both in mode m_1 and hence there is no switch. The following gives the modified MILP formulation:

Minimise

$$\underbrace{\sum_{k=1}^{|\mathcal{T}|} \sum_{s\in\mathcal{M}} \sum_{m\in\mathcal{M}} (C_{eff_k} P_{dnom_m} t_{k,s,m} + P_{leak_m} t_{k,s,m})}_{\text{task energy dissipation}} + \underbrace{\sum_{k=1}^{|\mathcal{T}|} \sum_{s\in\mathcal{M}} \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{M}} (b_{k,s,i,j}\, EP_{i,j})}_{\text{transition energy overhead}} \tag{20}$$

subject to:

$$\delta_k = \sum_{s\in\mathcal{M}^*} \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{M}} b_{k,s,i,j} DP_{i,j} \tag{21}$$

$$\delta_{k,l} = \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{M}} b_{k,m,i,j} DP_{i,j} \quad \text{where } (k,l) \in \mathcal{E}^{\bullet} \tag{22}$$

$$D_k + \sum_{s\in\mathcal{M}} \sum_{m\in\mathcal{M}} t_{k,s,m} + \delta_k \le dl_k \tag{23}$$

$$D_k + \sum_{s\in\mathcal{M}} \sum_{m\in\mathcal{M}} t_{k,s,m} + \delta_k + \delta_{pl,l} \le D_l$$

$$\forall (k,l) \in \mathcal{E}, (pl,l) \in \mathcal{E}^{\bullet} \tag{24}$$

$$c_{k,s,i} = t_{k,s,i} f_i \quad k \text{ in } 1, \ldots, |\mathcal{T}|,\ s \in \mathcal{M},\ i \in \mathcal{M},\ c \in \mathbb{N} \tag{25}$$

$$\sum_{s\in\mathcal{M}} \sum_{i\in\mathcal{M}} c_{k,s,i} = NC_k \quad k \text{ in } 1, \ldots, n \tag{26}$$

To capture the energy overheads in the objective function, (20), we introduce the Boolean variables $b_{k,s,i,j}$. In addition, we introduce an energy penalty matrix EP, which contains the energy overheads for all possible mode transitions, i.e. $EP_{i,j}$ denotes the energy overhead necessary to change form mode i to j. These energy overheads are precomputed based on the available modes (voltage pairs) and (4). The overall energy overhead is given by all intratask and intertask transitions. The intratask and intertask delay overheads, given in (21) and (22), are calculated based on a delay penalty matrix $DP_{i,j}$, which, similar to the energy penalty matrix, can be precomputed based on the available modes and (5). For a task τ_k and for each of its subtasks τ_k^s, except the last one, the variable $b_{k,s,i,j} = 1$ if mode i of subtask τ_k^s and mode j of τ_k^{s+1} are both active (s in$1, \ldots, |\mathcal{M}| - 1$, i,j in $1, \ldots, m$). These are used to capture the intratask overheads, as in (21). For intertask overheads, we are interested in the last mode of task τ_k and the first mode of the subsequent task τ_l (running on the same processor). Therefore, $b_{k,m,i,j} = 1$ if the mode i of the last subtask τ_k^m and the mode j of the first subtask τ_l^1 are both active. For the example given in Fig. 4*c*, $b_{1,1,2,1}$, $b_{1,2,1,1}$, $b_{1,3,1,3}$, $b_{2,1,3,3}$, $b_{2,2,3,2}$ are all 1 and the rest are 0. Deadlines and precedence relations, taking the delay overheads into account, have to be respected according to (23) and (24). Here, $\sum_{s\in\mathcal{M}} \sum_{m\in\mathcal{M}} t_{k,s,m}$ represents the total execution time of a task τ_k, based on the number of cycles in each of the subtasks and modes. Equations (25) and (26) are a reformulation of (18), which expresses the relation between the execution time and the number of clock cycles and the requirement to execute all clock cycles of a task. To ease the

explanation, the above given MILP formulation has been simplified to a certain degree. In particular, we have omitted here details on the computation of the b variables as well as the constraints that make sure that one and only one mode must be used by a subtask. The complete MILP model can be found in the Appendix.

7 Discrete voltage scaling heuristic

In the preceding Section, we have demonstrated that discrete voltage scaling is NP-hard and that solving it, using the MILP formulation, is time-consuming. Therefore, we propose the following heuristic to effectively solve the discrete voltage scaling problem. The main idea behind this heuristic is to perform a continuous voltage scaling (as outlined in Section 5.2) and to transform the continuously selected voltages into the discrete voltages of the processor. According to the operational frequencies that are calculated as a result of the continuously selected voltages, the two surrounding discrete performance modes are chosen, $f_{d1} < f_{con} < f_{d2}$. That is, the execution of a task is split into two regions with t_{d1} and t_{d2} being the execution times in mode with f_{d1} and f_{d2}, respectively. Figures 5*a* and 5*b* indicate this transformation for an application with three tasks. In the continuous scaling case, Fig. 5*a*, each of the tasks executes at a single voltage setting, i.e. the voltages are changed only between tasks. In the discrete case, the voltage setting is changed during the task execution. Of course, the required time overhead δ for the mode change has to be considered as well, i.e. $t = t_{d1} + t_{d2} + \delta$, where t is the task execution time with continuous voltage setting. In general, executing activities in two performance modes leads to close to optimal discrete voltage scaling [22]. Furthermore, restricting the execution to two settings avoids unnecessary intra-task transitions, which cause energy and time overheads. Having determined the discrete performance mode settings, the inter-task transition overheads are reduced by reordering the mode sequence of each task. From a task execution point of view, mode reordering does not have any effect. That is, if a certain task requires 600 cycles to execute, then it is equivalent to first executing 100 cycles at a lower voltage and then 500 at a higher voltage, or to first execute 500 cycles at a higher voltage and then 100 cycles at a lower voltage. We reorder the modes in such a way that a task starts execution with its execution in the lower (higher) performance mode if the preceding tasks on the same component finish execution in the lower (higher) performance mode. This is outlined in Fig. 5*c*. It is worthwhile to mention that this reordering only affects the order in which the voltages are altered, i.e. the code execution order inside the task remains unchanged. While this reordering technique is optimal for components that offer two performance modes, this is not true for components with three or more modes. Nevertheless, as demonstrated by our experiments, this heuristic is fast and efficient. Of course, the additional slack produced as a result of the reduced transition times is exploited as well.

8 Experimental results

We have conducted a set of experiments using numerous generated benchmarks as well as a real-life MPEG encoder example, to demonstrate the applicability of the presented approaches. The automatically generated benchmarks consist of 75 task graphs containing between 10 and 150 tasks, which are mapped and scheduled onto architectures composed of 1 to 3 processors. The technology dependent parameters of these processors were considered to correspond to a CMOS fabrication in 70 nm, for which the leakage power represents ~50% of the total power consumed. For experimental purposes the amount of deadline slack in each benchmark was varied over a range 0 to 90%, using a 10% increment, resulting in 750 performed evaluations, carried out with the aim to achieve representative average values.

8.1 Supply voltage scaling vs combined supply and body-bias voltage scaling

The first set of experiments was conducted to demonstrate the achievable energy savings when comparing the classic V_{dd} selection with simultaneous V_{dd} and V_{bs} selection. Figure 6*a* shows the outcomes for the continuous voltage selection with and without the consideration of transition overheads. The continuous voltage ranges were set to $0.6\text{V} \leq V_{dd} \leq 1.8\text{V}$ and $-1\text{V} \leq V_{bs} \leq 0$. The values for C_r, C_s, p_{Vdd} and p_{Vbs} were set to 10 µF, 40 µF, 100 µs/V, and 100 µs/V, respectively. The Figure shows the percentage of total energy consumed (relative to the baseline energy) as a function of the available slack within the application. As a baseline we consider the energy consumption at the nominal (highest) voltage for V_{dd} and V_{bs}. It is easy to observe the advantage of the combined voltage selection scheme over the classical voltage selection, with a difference of up to 40%. These observations hold with and without the consideration of overheads. Regarding the overhead influence on the overall energy consumption, we can see from the Figure that the savings are ~1% for the combined scheme and 2% for the classical voltage selection. These moderate amounts of additional savings have a straightforward explanation: Within the continuous scheme (which from a practical point of view is unrealistic), the voltage differences between tasks are likely to be small, i.e. large overheads are avoided (see (4) and (5)).

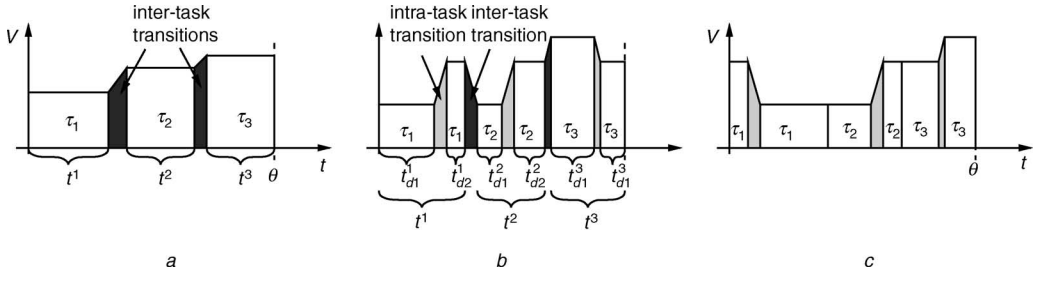

Fig. 5 *Performance mode reordering*

a Continuous voltage schedule (solely intertask mode transitions)
b Discrete voltage schedule (5 performance mode transitions)
c Reordered discrete schedule (3 performance mode transitions)

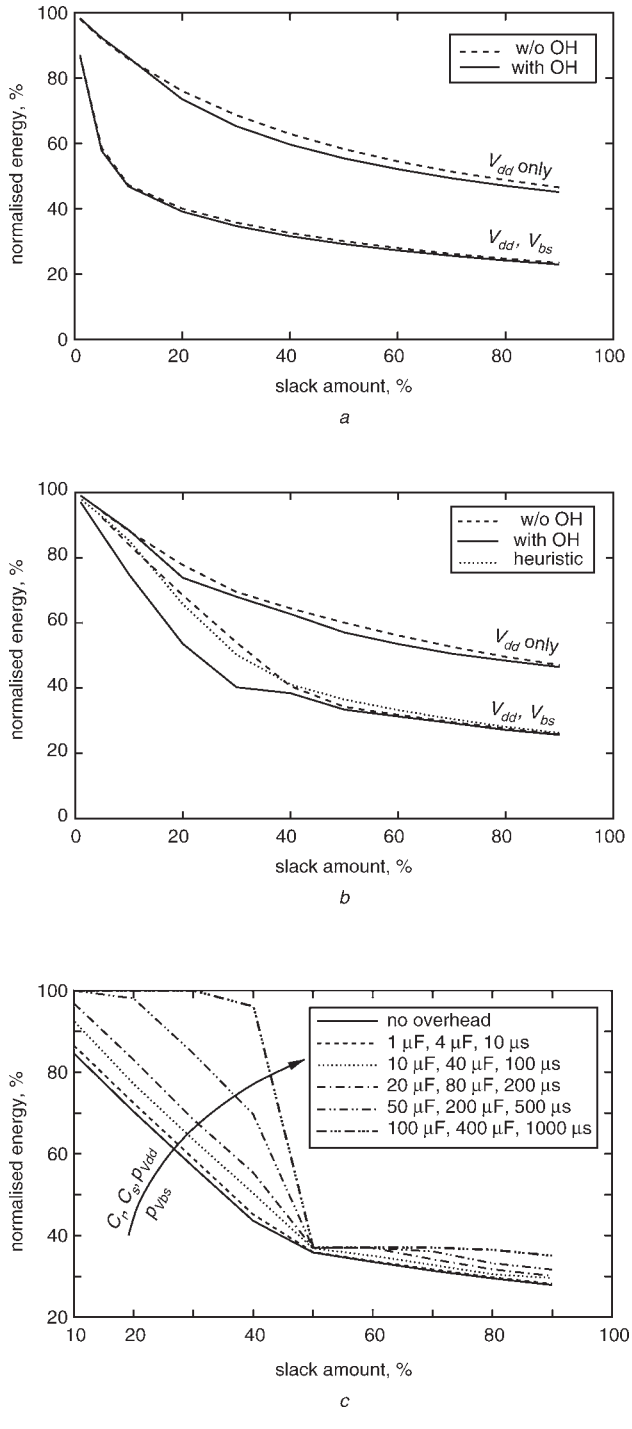

Fig. 6 *Optimisation results*

a Continuous voltage selection
b Discrete voltage selection
c Influence of scaling overheads

We have further evaluated the discrete voltage selection scheme. Here the processors could switch between three different voltage settings (1.8,0), $(1.5, -0.4)$ and $(1.2, -0.6)$ for the combined scheme, and 1.8, 1.5 and 1.2 for the classical V_{dd} selection. The results are given in Fig. 6*b*. As in the continuous case, we can observe the difference between the classical supply voltage selection and the more efficient combined selection scheme. For low amounts of slack ($\sim$10%), the savings for the combined selection are significantly lower than in the continuous case. The reason for this is that, due to the small slack available, the processors have to run in the highest voltage mode, which does not reduce leakage power. Further, we can see that, with increasing slack, the overall energy approaches the theoretical minimum given by the continuous case, since more time is spent in the energy-efficient mode m_3. It is interesting to observe the influence of the transition overheads, in particular when not much system slack is available. In this situation the unnecessary switching between voltages to exploit the 'small' amounts of slack causes an increased energy overhead. Consider, for instance, the cases where the combined V_{dd} and V_{bs} selection has been optimised with and without overheads. Between 10% to 40% of slack, the consideration of transition overheads results in improved solutions with up to 12% higher savings. Of course, with increasing slack the number of tasks executed at the lowest voltage setting increases, and hence the number of transitions is decreased. As a result, the influence of the transition overheads decreases. It should be noted that the reported results for the discrete scheme have been evaluated using graphs with at most 80 tasks (without overhead, DNOH) and 30 tasks (with overhead, DOH), since the required optimisation times become intractable, as a result of the NP-hardness of the problem (Section 6.1). To overcome this problem we have additionally investigated the proposed voltage selection heuristic. As outlined in Section 7, this heuristic uses the voltage schedules derived from the continuous selection (COH). For each selected continuous task voltage, the two surrounding discrete voltage pairs are chosen (similarly to the classical approach proposed in [1]). To minimise overheads, we perform a simple reordering of mode activations. The results of this simple heuristic follow the discrete voltage selection without overheads, as shown in Fig. 6*b*. However, due to its relatively reduced polynomial time complexity, it can be applied to large instances of the problem.

8.2 Significance of transition overheads

To further investigate the influence of transition overheads, we have carried out an additional set of experiments in which the amount of the processors' overheads in terms of energy and delay were varied by adjusting the values for C_r, C_s, p_{Vdd} and p_{Vbs} (see Section 2.2) In accordance, we use the discrete voltage selection with consideration of overheads. The results are given in Fig. 6*c*. As expected, the energy dissipation increases for higher values of the overhead determining parameters. For instance, while a 'hypothetical' processor which requires no transition overheads can reduce the energy consumption by 58% if 40% of slack is available, a realistic processor with $C_r = 20\,\mu\text{F}$, $C_s = 80\,\mu\text{F}$, $p_{Vdd} = 200\,\mu\text{s/V}$ and $p_{Vbs} = 200\,\mu\text{s/V}$ achieves only 42%. This highlights the importance to carefully consider the influence of transition overheads.

8.3 Real-life example

In addition to the above given benchmark results, we have conducted experiments on a real-life MPEG encoder application, to validate the real-world applicability of the presented techniques. Details regarding this application can be found in [20].

The MPEG encoder consists of 109 tasks and is considered to run on an architecture composed of 2 processing elements with 2 voltage modes $((1.8\,\text{V}, -0.2\,\text{V})$ and $(1.0\,\text{V}, -0.5))$. At the highest voltage mode, the application reveals a deadline slack close to 40%. Switching overheads are characterised by $C_r = 10\,\mu\text{F}$, $C_s = 40\,\mu\text{F}$, $p_{Vdd} = 10\,\mu\text{s/V}$, and $p_{Vbs} = 10\,\mu\text{s/V}$. Table 1 shows the resulting energy consumptions in terms of dynamic E_{dyn}, leakage E_{leak}, overhead ε and total E_Σ energy (columns 2–5).

Table 1: Optimisation results for MPEG encoder

Approach	E_{dyn}, µJ	E_{leak}, µJ	ε, µJ	E_{Σ}, µJ	Reduction, %
Nominal	12.64	11.1	non	23.74	–
DVDDNOH	9.03	8.33	0.44	17.80	25.0
DVDDOH	9.054	8.31	0.006	17.37	26.9
DNOH	9.92	4.05	0.80	14.77	37.8
DOH	9.91	4.05	0.02	13.98	41.1
Heuristic	10.21	4.20	0.09	14.50	39.0

Each line represents a different voltage selection approach. Line 2 (nominal) is used as a baseline and corresponds to an execution at the nominal voltages. Lines 3 and 4 give the results for the classical V_{dd} selection, without (DVDDNOH) and with (DVDDOH) the consideration of overheads. As we can see, the consideration of overheads achieves higher energy saving (26.9%) than the overhead neglecting optimisation (25.0%). Although the dynamic energy is slightly increased when considering the overheads, the total energy is minimised due to the reduction of transition overheads. The results given in lines 5 and 6 correspond to the combined V_{dd} and V_{bs} selection schemes. Again we distinguish between overheads neglecting (DNOH) and overhead considering (DOH) approaches. If the overheads are neglected, the energy consumption can be reduced by 37.8%, yet taking the overheads into account results in an reduction of 41.1%, solely achieved by decreasing the transition overheads. Compared to the classical voltage selection scheme (26.9% savings), the combined selection achieved a further reduction of 14.2%. These experiments underline how the consideration of transition overheads helps in achieving energy-efficient voltage schedules. For comparison, the last line shows the results of the heuristic approach. Although the result does not match the optimal one given in line 6, it should be noted that such heuristic techniques are needed when dealing with problems of larger complexity (increased number of voltage modes and tasks). In the MPEG application, although the number of tasks is realistically large, we considered only two voltage modes. Therefore the optimal solutions could be obtained for the DOH problem.

Overall, the conducted experiments have demonstrated the advantages of the combined voltage selection over the classical V_{dd} scheme. Furthermore, it was shown that the consideration of transition overheads has a profound impact on the overall achievable energy savings.

9 Conclusions

Energy reduction techniques, such as dynamic voltage scaling and adaptive body biasing can be effectively exploited at the system level In this paper, we have investigated different notions of the combined dynamic voltage scaling and adaptive body-biasing problem at the system-level. These include the consideration of transition overheads as well as the discretisation of the supply and threshold voltage levels. It was demonstrated that nonlinear programming and mixed integer linear programming formulations can be used to solve these problems. Further, the NP-hardness of the discrete voltage scaling case was shown, and a heuristic to efficiently solve the problem has been proposed. Several generated benchmark examples as well as a real-life voice codec example were used to show the applicability of the introduced approaches.

10 References

1 Ishihara, T., and Yasuura, H.: 'Voltage scheduling problem for dynamically variable voltage processors'. Proc. Int. Symp. on Low Power Electronics and Design (ISLPED), 1998, pp. 197–202
2 Martin, S., Flautner, K., Mudge, T., and Blaauw, D.: 'Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads'. Proc. ICCAD, 2002, pp. 721–725
3 Yao, F., Demers, A., and Shenker, S.: 'A scheduling model for reduced CPU energy'. Proc. IEEE Annual Foundations of Computer Science, 1995, pp. 374–382
4 Kim, C., and Roy, K.: 'Dynamic Vth scaling scheme for active leakage power reduction'. Proc. Conf. on Design, Automation and Test in Europe (DATE), March 2002, pp. 163–167
5 Borkar, S.: 'Design challenges of technology scaling', *IEEE Micro*, 1999, **19**, (4), pp. 23–29
6 Kwon, W., and Kim, T.: 'Optimal voltage allocation techniques for dynamically variable voltage processors'. Proc. IEEE DAC, June 2003, pp. 125–130
7 Gruian, F., and Kuchcinski, K.: 'LEneS: Task scheduling for low--energy systems using variable supply voltage processors'. Proc. ASP-DAC, Jan 2001, pp. 449–455
8 Luo, J., and Jha, N.: 'Power-profile driven variable voltage scaling for heterogeneous distributed real-time embedded systems'. Proc. VLSI, 2003
9 Schmitz, M.T., and Al-Hashimi, B.M.: 'Considering power variations of DVS Processing Elements for energy minimisation in distributed systems'. Proc. Int. Symp. on System Synthesis (ISSS), October 2001, pp. 250–255
10 Zhang, Y., Hu, X., and Chen, D.: 'Task scheduling and voltage selection for energy minimization'. Proc. IEEE DAC, June 2002
11 Duarte, D., Vijaykrishnan, N., Irwin, M., Kim, H., and McFarland, G.: 'Impact of scaling on the effectiveness of dynamic power reduction'. Proc. ICCD, Sept. 2002
12 Hong, I., Qu, G., Potkonjak, M., and Srivastava, M.B.: 'Synthesis techniques for low-power hard real-time systems on variable voltage processors'. Proc. Symp. on Real-Time Systems, 1998
13 Mochocki, B., Hu, X., and Quan, G.: 'A realistic variable voltage scheduling model for real-time applications'. Proc. ICCAD, 2002, pp. 726–731
14 Zhang, Y., Hu, X., and Chen, D.: 'Energy minimization of real-time tasks on variable voltage processors with transition energy overhead'. Proc. ASP-DAC, 2003, pp. 65–70
15 Yan, L., Luo, J., and Jha, N.: 'Combined dynamic voltage scaling and adaptive body biasing for heterogeneous distributed real-time embedded systems'. Proc. ICCAD, 2003
16 Chandrakasan, A.P., and Brodersen, R.W.: 'Low power digital CMOS design' (Kluwer Academic Publisher, 1995)
17 Nesterov, Y., and Nemirovskii, A.: 'Interior-point polynomial algorithms in convex programming' (SIAM, 1994)
18 Intel® XScale™ Core, Developer's Manual, December 2000
19 Mobile AMD Athlon™4, Processor Model 6 CPGA Data Sheet, November 2000. Publication No 24319 Rev E
20 Andrei, A., Schmitz, M.T., Eles, P., and Peng, Z.: 'Overhead-conscious voltage selection for Dynamic and leakage energy reduction of time-constrained systems. Technical report, Linkoping University, Department of Computer and Information Science, Sweden, September 2003
21 De, P., Dunne, E., Ghosh, J., and Wells, C.: 'Complexity of the discrete time-cost tradeoff problem for project networks', *Oper. Res.*, 1997, **45**, (2), pp. 302–306
22 Andrei, A., Schmitz, M., Eles, P., Peng, Z., and Al-Hashimi, B.: 'Overhead-conscious voltage selection for dynamic and leakage power reduction of time-constraint Systems'. Proc. Conf. on Design, Automation and Test in Europe (DATE), Feb. 2004, pp. 518–523

11 Appendix: Complete discrete voltage selection with overheads MILP formulation

This Appendix outlines in more detail the MILP formulation for the discrete voltage selection problem, i.e. it provides additional information that has been withheld from Section 6.3 for clarity reasons. The complete formulation is given by:

Minimise

$$\underbrace{\sum_{k=1}^{|\mathcal{T}|}\sum_{s\in\mathcal{M}}\sum_{m\in\mathcal{M}}\left(C_{eff_k}P_{dnom_m}t_{k,s,m}+P_{leak_m}t_{k,s,m}\right)}_{\text{task energy dissipation}} + \underbrace{\sum_{k=1}^{|\mathcal{T}|}\sum_{s\in\mathcal{M}}\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{M}}\left(b_{k,s,i,j}\,EP_{i,j}\right)}_{\text{transition energy overhead}} \quad (27)$$

Subject to:

$$\delta_k = \sum_{s\in\mathcal{M}^*}\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{M}} b_{k,s,i,j}\, DP_{i,j} \tag{28}$$

$$\delta_{k,l} = \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{M}} b_{k,m,i,j}\, DP_{i,j} \quad \text{where } (k,l)\in\mathcal{E}^{\bullet} \tag{29}$$

$$D_k + \sum_{s\in\mathcal{M}}\sum_{m\in\mathcal{M}} t_{k,s,m} + \delta_k \le dl_k \tag{30}$$

$$D_k + \sum_{s\in\mathcal{M}}\sum_{m\in\mathcal{M}} t_{k,s,m} + \delta_k + \delta_{pl,l} \le D_l$$
$$\forall (k,l)\in\mathcal{E}, (pl,l)\in\mathcal{E}^{\bullet} \tag{31}$$

$$c_{k,s,i} = t_{k,s,i}\, f_i \quad k \text{ in } 1,\ldots,|\mathcal{T}|,\ s\in\mathcal{M},\ i\in\mathcal{M},\ c\in\mathbb{N} \tag{32}$$

$$\sum_{s\in\mathcal{M}}\sum_{i\in\mathcal{M}} c_{k,s,i} = NC_k \quad k \text{ in } 1,\ldots,n \tag{33}$$

Up to this point the model corresponds to the formulation given in Section 6.3. The additional constraints that complete the formulation are:

$$\sum_{m\in\mathcal{M}} c_{k,s,m} \ge 1 \quad \forall \tau_k,\ s\in\mathcal{M} \tag{34}$$

$$a_{k,s,m}\, NC_k \ge c_{k,s,m} \quad k\in 1,\ldots,|\mathcal{T}|,\ s\in\mathcal{M}, m\in\mathcal{M} \tag{35}$$

$$\sum_{m\in\mathcal{M}} a_{k,s,m} = 1 \quad k\in 1,\ldots,|\mathcal{T}|, s\in\mathcal{M} \tag{36}$$

$$\begin{cases} a_{k,s,i} = \sum_{j=1}^{m} b_{k,s,i,j} \\ \quad k\in 1,\ldots,|\mathcal{T}|, s \text{ in } 1,\ldots,|\mathcal{M}|-1, i\in\mathcal{M} \\ a_{k,s+1,j} = \sum_{i=1}^{m} b_{k,s,i,j} \\ \quad k\in 1,\ldots,|\mathcal{T}|, s \text{ in } 2,\ldots,|\mathcal{M}|, j\in\mathcal{M} \end{cases} \forall (k,l)\in E \tag{37}$$

$$\begin{cases} a_{k,s,i} = \sum_{j=1}^{m} b_{k,m,i,j} & s\in\mathcal{M},\ i\in\mathcal{M} \\ a_{l,1,j} = \sum_{i=1}^{m} b_{k,m,i,j} & j\in\mathcal{M} \end{cases} \forall (k,l)\in E \tag{38}$$

$$c_{k,s,m}\in\mathbb{N},\ a_{k,s,m},\ b_{k,s,i,j}\in\{0,1\}$$
$$k = 1,\ldots,|\mathcal{T}|,\ s\in\mathcal{M},\ i\in\mathcal{M}, j\in\mathcal{M} \tag{39}$$

In Section 6.3, we have briefly introduced the MILP model with the transition overheads. We detail now how we capture the mode variations in our MILP formulation. Please remember that to compute the corresponding delay and energy penalties, the concepts of subtasks and execution modes have been introduced in Section 6.3. Equation (34) states that for each subtask τ_k^s of a task τ_k, at least one mode m is active (at least one clock cycle is executed by a subtask). This can be oberseved, for example, in Fig. 4*c*.

Further, we introduced two sets of auxiliary variables: $a_{k,s,m}$ and $b_{k,m,i,j}$. The binary variables $a_{k,s,m}$ indicate, for a given task τ_k, the active mode m for each subtask τ_k^s. $a_{k,s,m}$ is 1 when $c_{k,s,m} \ge 1$ and 0 when $c_{k,s,m} = 0$. For instance, Fig. 4*d* gives the binary variables $a_{k,s,m}$ for the solution vector in Fig. 4*c*. The other binary variables, $b_{k,s,i,j}$, are the instrument directly used to compute the penalties, both in terms of energy and delay. For all tasks τ_k, a mode change from subtask τ_k^s with mode i to subtask τ_k^{s+1} with mode j (s in $1,\ldots,|\mathcal{M}|-1$) is expressed by $b_{k,s,i,j} = 1$. Otherwise, i.e. in the case of no mode change, $b_{k,s,i,j} = 0$. The binary variables $b_{k,s,i,j}$ are used in (27), (28) and (29), the equations in which the energy and delay overheads are computed. Please note that (34) to (39) are solely introduced to pinpoint where mode changes are situated.