Energy Optimization of Multiprocessor Systems on Chip by Voltage Selection

Alexandru Andrei, Petru Eles Zebo Peng
Linköping University
SE-58 183 Linköping, Sweden
Marcus Schmitz, Bashir M. Al-Hashimi
University of Southampton
Southampton, SO17 1BJ, United Kingdom

***Abstract*— Dynamic voltage selection and adaptive body biasing have been shown to reduce dynamic and leakage power consumption effectively. In this paper, we optimally solve the combined supply voltage and body bias selection problem for multi-processor systems with imposed time constraints, explicitly taking into account the transition overheads implied by changing voltage levels. Both energy and time overheads are considered. The voltage selection technique achieves energy efficiency by simultaneously scaling the supply and body bias voltages in the case of processors and buses with repeaters, while energy efficiency on fat wires is achieved through dynamic voltage swing scaling. We investigate the continuous voltage selection as well as its discrete counterpart, and we prove strong NP-hardness in the discrete case. Furthermore, the continuous voltage selection problem is solved using nonlinear programming with polynomial time complexity, while for the discrete problem we use mixed integer linear programming and a polynomial time heuristic. We propose an approach that combines voltage selection and processor shutdown in order to optimize the total energy.**

I. Introduction

Embedded computing systems need to be energy efficient, yet they have to deliver adequate performance to computational expensive applications, such as voice processing and multimedia. The workload imposed on such an embedded system is non-uniform 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 trade-off during run-time of the application are dynamic voltage selection (DVS) [1], [2], [3] and adaptive body biasing (ABB) [4], [2]. 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. Up to date, most research efforts at the system-level were 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 (in order to maintain peak performance) is resulting in the fact that a substantial portion of the overall power dissipation will be due to leakage currents [5], [4]. This makes the adaptive body-biasing approach and its combination with dynamic voltage selection attractive for energy-efficient designs in the foreseeable future.

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 at design time and then applied at run-time according to the pre-calculated voltage schedule.

There has been a considerable amount of work on dynamic voltage selection. Yao et al. [3] proposed the first DVS approach for single processor systems which can change the supply voltage over a continuous range. Ishihara and Yasuura [1] modeled 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 non-uniform switched capacitance. Although this work gives the impression that the 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 selection has also been successfully applied to heterogeneous distributed systems, mostly using heuristics [7], [8], [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 only the supply voltage V_{dd} 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] analyze the effectiveness of supply and threshold voltage selection, and show that simultaneously adjusting both voltages provides the highest savings. Martin et al. [2] presented an approach for combined dynamic voltage selection and adaptive body-biasing. At this point we should emphasize 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 multi-processor 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. Noticeable exceptions are [12], [13], [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 minimize dynamic as well as leakage energy. In particular, we investigate four different notions of the combined dynamic voltage selection 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 combination of dynamic supply voltage selection and processor shutdown was presented in [16] for single processor systems. The authors demostrate the existence of a critical speed, under which scaling the processor frequency becomes energy inefficient, due to the fact that the leakage energy increases faster than the dynamic energy decreases. The leakage energy reduction is achieved there by shutting down the processor during the idle intervals, without performing adaptive body biasing.

To fully exploit the potential performance provided by multiprocessor architectures (e.g. systems-on-a-chip), communication has to take place over high performance buses, which interconnect the individual components, in order to prevent performance degradation through unnecessary contention. Such global buses require a substantial portion of energy, on top of the energy dissipated by the computational components [17], [18]. The minimization of the overall energy consumption requires the combined optimization of both the energy dissipated by the computational processors as well as the energy consumed by the interconnection infrastructure.

A negative side-effect of the shrinking feature sizes is the increasing *RC* delay of on-chip wiring [19], [18]. The main reason behind this trend is the ever-increasing line resistance. In order to maintain high performance it becomes necessary to "speed-up" the interconnects. Two implementation styles which can be applied to reduce the propagation delay are: (a) The insertion of *repeaters* and (b) the usage of *fat wires*. In principle, repeaters split long wires into shorter (faster) segments [19], [20], [18] and fat wires reduce the wire resistance [17], [18]. Techniques for the determination of the optimal quantity of repeaters are introduced in [19], [20]. An approach to calculate the optimal voltage swing on fat wires has been proposed in [17]. Similar to processors with supply voltage selection capability, approaches for link voltage scaling were presented in [21], [22]. An approach for communication speed selection was outlined in [23]. Another possibility to reduce communication energy is the usage of bus encoding techniques [24]. In [25], it was demonstrated that shared-bus splitting, which dynamically breaks down long, global buses into smaller, local segments, also helps to improve energy savings. An estimation framework for communication switching activity was introduced in [26].

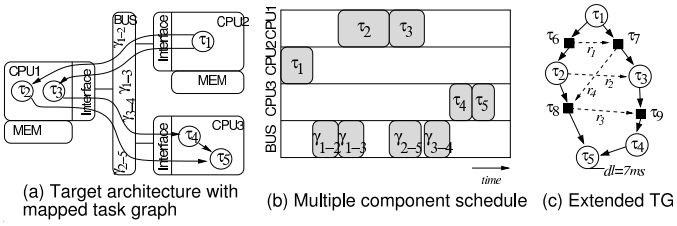
Until now, energy estimation for system-level communication was treated in a largely simplified manner, [23], [27], and based on naive models that ignore essential aspects such as bus implementation technique (repeaters, fat wires), leakage power, and voltage swing adaption. This, however, very often leads to oversimplifications which affect the correctness and relevance of the proposed approaches and, consequently, the accuracy of results. On the other hand, issues like optimal voltage swing and increased leakage power due to repeaters are not considered at all for implementations of voltage-scalable embedded systems. We have presented preliminary results regarding processor voltage selection and simulataneous processor and communication voltage selection in [28] and [29].

Our contributions are the following:

(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 multi-processor 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 $\varepsilon > 0$).
(d) We solve the combined voltage selection problem for processing elements and communications links. To allow an effective voltage selection on the communication links, we outline a set of delay and energy models. Further, we take into account the possibility of dynamic voltage swing scaling on fat wires and address the leakage power dissipation in bus repeaters.
(e) Since voltage selection for components that operate with discrete voltages is proofed to be NP-hard, we introduce a simple yet effective heuristic based on the NLP formulation for the continuous voltage selection problem.
(f) We study the combined voltage selection and processor shutdown problem. In particular, we demonstrate that the processor shutdown is an NP complete problem even isolated from the voltage selection. We propose two solutions that integrate the shutdown with the continuous and respectively with the discrete voltage selection.

As mentioned earlier, in this paper we will concentrate on off-line voltage selection techniques, that make use of the static slack existing in the application. In [30] we presented an efficient technique that dynamically makes use of slack created online, due to the fact that tasks execute less then their worst case number of clock cycles. Although the details of that technique are beyond the scope of this paper, in section X we will briefly introduce its principles and illustrate its effectiveness in conjunction with the shutdown procedure.

The remainder of this paper is organized as follows: Preliminaries regarding the system specification, the processor power and delay models are given in Sections II and III. This is followed by a motivational example in Section IV. The four investigated processor voltage selection problems are formulated in Section V. Continuous and discrete voltage selection problems are discussed in Sections VI and VII, respectively. We study the combined voltage selection and shutdown problem in Section VIII. Power and delay models for the communication links are given and the general problem



(a) Target architecture with mapped task graph (b) Multiple component schedule (c) Extended TG

Fig. 1. System models

of voltage selection for processors and the communication is addressed in Section IX. Extensive experimental results are presented in Section X, and conclusions are drawn in Section XI.

II. System and Application Model

In this paper we consider embedded systems which are realized 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 applications is captured by task graphs $G(\Pi,\Gamma)$. Nodes $\tau \in \Pi$ in these directed acyclic graphs represent computational tasks, while edges $\gamma \in \Gamma$ indicate data dependencies between these tasks (communications). Tasks τ_i require in the worst case NC_i clock cycles to be executed, depending on the PE to which they are mapped. Further, tasks are annotated with deadlines dl_i that have to be met at run-time.

If two dependent tasks are assigned to different PEs, p_x and p_y with $x \neq y$, then the communication takes place over a CL, involving a certain amount of time and power.

We assume that the task graph is mapped and scheduled on the target architecture, i.e., it is known where and in which order tasks and communications take place. Fig. 1(a) shows an example task graph that has been mapped onto an architecture and Fig. 1(b) depicts a possible execution order.

To tie the execution order into the application model, we perform the following transformation on the original task graph. First, all communications that take place over communication links are captured by communication tasks, as indicated by squares in Fig. 1(c). For instance, communication γ_{1-2} is replaced by task τ_6 and the edges connecting τ_6 to τ_1 and τ_2 are introduced. $\mathcal{K}$ defines the set of all such communication tasks and $\mathcal{C}$ the set of graph edges obtained after the introduction of the communication tasks. Furthermore, we denote with $\mathcal{T} = \Pi \cup \mathcal{K}$ the set of all computations and communications. Second, 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 $\mathcal{E} = \mathcal{C} \cup \mathcal{R}$. We construct the mapped and scheduled task graph $G(\mathcal{T},\mathcal{E})$. 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 or CL.

III. Processor 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 [31], [2],

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

where C_{eff}, f, and V_{dd} denote the effective charged capacitance, operational frequency, and circuit supply voltage, respectively. Although, until recently, 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 ($< 65nm$) it is expected that leakage will account for a significant part of the total power. The leakage power is given by [2],

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

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 stronger influenced by V_{bs} than by V_{dd}, due to the fact that the constant K_5 is larger than the constant K_4 (e.g., for the Crusoe processor described in [2], $K_5 = 4.19$ while $K_4 = 1.83$).

Nevertheless, scaling the supply and the body-bias voltage for power saving, has a side-effect on the circuit delay d and hence the operational frequency [31], [2]:

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

where α reflects the velocity saturation imposed by the used technology (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 often is overlooked, is the consideration of transition overheads, i.e., each time the processor's supply 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],

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

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

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

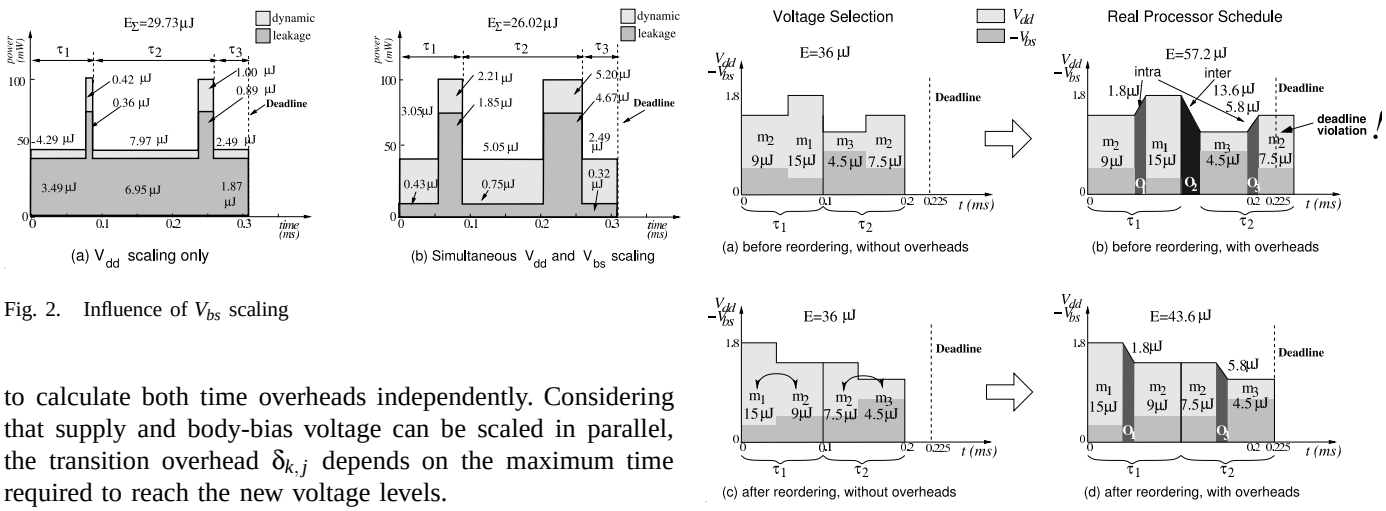


Fig. 2. Influence of V_{bs} scaling

Fig. 3. Influence of transition overheads

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.

In the following, we assume that the processors can operate in several execution modes. An execution mode m_z is characterized by a pair of supply and body bias voltages: $m_z = (V_{dd_z}, V_{bs_z})$. As a result, an execution mode has an associated frequency and power consumption (dynamic and leakage) that can be calculated using Eq. 3 and respectively Eq. 1 and 2. Upon a mode change, the corresponding delay and energy penalties are computed using Eq. 5 and 4.

Tasks that are mapped on different processors communicate over one or more shared buses. In sections 4-8 we assume that the buses are not voltage scalable and thus working at a given frequency. Each communication task has a fixed execution time and energy consumption depeding proportionally on the amount of communication. For simplicity of the explanations, in sections 4-8 we will not differentiate between computation and communication tasks. A more refined communication model, as well as the benefits of simultaneously scaling the voltages of the processors and communication links is introduced in Section IX.

IV. Motivational Examples

A. Optimizing the Dynamic and Leakage Energy

Fig. 2 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. For simplicity we neglect transition overheads in this example. Further, we consider processor parameters that correspond to CMOS technology ($< 70nm$) 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 $V_{dd1} = 1.8V$, or $V_{dd2} = 1.5V$, while V_{bs1} and V_{bs2} are kept at $0V$. In accordance, the system dissipates $P_{dyn1} = 100mW$ and $P_{leak1} = 75mW$ in mode 1 running at $700MHz$, while $P_{dyn2} = 49mW$ and $P_{leak2} = 45mW$ in mode 2 running at $525MHz$, 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.

The total leakage and dynamic energies of the schedule in Fig. 2(a) are $13.56\mu J$ and $16.17\mu J$, respectively. This results in a total energy consumption of $29.73\mu J$.

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

B. Considering the Transition Overheads

We consider a single processor system that offers three voltage modes, $m_1 = (1.8V, -0.3V)$, $m_2 = (1.5V, -0.45V)$, and $m_3 = (1.2V, -0.8V)$, where $m_z = (V_{dd_z}, V_{bs_z})$. The rail and substrate capacitance are given as $C_r = 10\mu F$ and $C_s = 40\mu F$. The processor needs to execute two consecutive tasks (τ_1 and τ_2) with a deadline of $0.225ms$. Fig. 3(a) shows a possible voltage schedule. 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 $E = 9 + 15 + 4.5 + 7.5 = 36\mu J$. However, if this voltage schedule is applied to a *real* voltage-scalable processor, the resulting schedule will be affected by transition overheads, as shown in Fig. 3(b). The processor requires a given time to adapt to the new execution mode. During this adaption no computations can be performed [32], [33], which increases the schedule length such that the imposed deadline is violated. Moreover, transitions do 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 $10\mu F \cdot (1.8V - 1.5V)^2 + 40\mu F \cdot (0.3V - 0.45V)^2 = 1.8\mu J$, based on Eq. (4). Similarly, the energy overheads for transitions O_2 and O_3 can be calculated as $13.6\mu J$ and $5.8\mu J$, respectively. The overall energy dissipation of the schedule from Fig. 3(b) accumulates to $36 + 1.8 + 13.6 + 5.8 = 57.2\mu J$.

Compared to the schedule in Fig. 3(a), the mode activation order in Fig. 3(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. 3(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 (inter switch) has been eliminated. The overall energy consumption has been reduced to $E = 43.6\mu J$, a reduction by 23.8% compared to the schedule given in Fig. 3(b). Further, the elimination of transition O_2 reduces the overall schedule length, such that the imposed deadline is satisfied. With this example we have illustrated the effects that transition overheads can have on the energy consumption and the timing behavior and the impact of taking them into consideration when elaborating the voltage schedule.

V. Problem Formulation

Consider a set of tasks $\mathcal{T} = \{\tau_i\}$ with precedence constraints, that have been mapped and scheduled on a set of variable voltage processors. For each task τ_i its deadline dl_i, its worst case 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 voltage pairs $m_z = \{(V_{dd_z}, V_{bs_z})\}$ (for the discrete problem). The power dissipations (leakage and 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 Eqs. (4) and (5)).

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

VI. Optimal Continuous Voltage Selection

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

A. Continuous Voltage Selection without Overheads (CNOH)

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

$$\sum_{k=1}^{|\mathcal{T}|} \Big(\underbrace{NC_k \cdot C_{eff_k} \cdot V_{dd_k}^2}_{E_{dyn_k}} + \underbrace{L_g (K_3 \cdot V_{dd_k} \cdot e^{K_4 \cdot V_{dd_k}} \cdot e^{K_5 \cdot V_{bs_k}} + I_{Ju} \cdot |V_{bs_k}|) \cdot t_k}_{E_{leak_k}} \Big) \quad (6)$$

subject to

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

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

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

$$D_k \geq 0 \quad (10)$$

$$V_{dd_{min}} \leq V_{dd_k} \leq V_{dd_{max}} \text{ and } V_{bs_{min}} \leq V_{bs_k} \leq V_{bs_{max}} \quad (11)$$

The variables that need to be determined 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 total energy consumption, which is the sum of dynamic and leakage energy, has to be minimized, as in Eq. (6)[1]. 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 Eq. (7). Given the execution time of the tasks, it becomes possible to express the precedence constraints between tasks (Eq. (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 $(k,l) \in \mathcal{E}$ in the mapped and scheduled task graph. Similarly, tasks with deadlines have to be completed ($D_k + t_k$) before their deadlines dl_k (Eq. (9)). Task start times have to be positive (Eq. (10)) and the imposed voltage ranges should be respected (Eq. (11)). It should be noted that the objective (Eq. (6)) as well as the task execution time (Eq. (7)) are convex functions. Hence, the problem falls into the class of general convex nonlinear optimization problems. Such problems can be efficiently solved in polynomial time (given an arbitrary precision $\varepsilon > 0$), [35].

[1]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 [34].

B. 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.

Minimize

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

subject to

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

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

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

VII. Optimal Discrete Voltage Selection

The approaches presented in the previous section provide a theoretical upper 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 analyze the complexity of the discrete voltage selection problem.

A. Problem Complexity

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

Proof: We proof by restriction. The discrete time-cost trade-off (DTCT) problem is known to be NP-hard [36]. 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 details of the proof are given in [34]. The problem is NP-hard, even if restricted it to supply voltage selection (without adaptive body-biasing) and even if transition overheads are neglected. It should be noted that this finding renders the conclusion of [6][2] impossible, which states that the discrete voltage selection problem (considered in [6] without body-biasing and overheads) can be solved optimally in polynomial time.

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

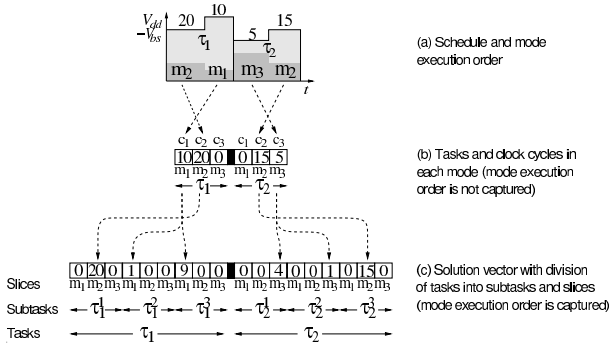


Fig. 4. Discrete mode model

B. Discrete Voltage Selection without Overheads (DNOH)

In the following we will 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 characterized by a voltage pair (V_{dd_m}, V_{bs_m}), which determines the operational frequency f_m, the normalized dynamic power P_{dnom_m}, and the leakage power dissipation P_{leak_m}. The frequency and the leakage power are given by Eqs. (3) and (2), respectively. The normalized dynamic power is given by $P_{dnom_m} = f_m \cdot V_{dd_m}^2$. Accordingly, the dynamic power of a task τ_k operating in mode m is computed as $C_{eff_k}\cdot P_{dnom_m}$. Based on these definitions, the problem is formulated as follows:
Minimize

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

subject to

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

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

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

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

The total energy consumption, expressed by Eq. (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. The outer sum adds up the energy of all tasks. Unlike the continuous voltage selection 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. The number of clock cycles $c_{k,m}$ is restricted to the integer domain. We exemplify this model graphically in Figures 4(a) and 4(b). The first figure shows the schedule of two tasks

executing each at two different voltage settings (two modes out of three possible). 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 runtime have the corresponding time and number of clock cycles 0 (mode m_3 for τ_1 and $m1$ for τ_2). The overall execution time of task τ_k is given as the sum of the times spent in each mode ($\sum_{m\in\mathcal{M}} t_{k,m}$). Eq. (16) ensures that all the deadlines are met and Eq. (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 Eq. (18). Additionally, task start times D_k and task execution times have to be positive (Eq. (19)).

C. Discrete Voltage Selection with Overheads (DOH)

We now proceed with the incorporation of transition overheads into the MILP formulation given in Section VII-B. The order in which the modes are activated has an influence on the transition overheads, as we have illustrated in Section IV-B. Nevertheless, the formulation in Section VII-B does not capture the order in which modes are activated, it solely expresses how many clock cycles are spent in each mode. 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 $\tau_k^s, s = 1,...,m$. 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 $m \cdot m$ slices for each task. Fig. 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 the mode m_2. Note that there is no overhead between subsequent subtasks that run in the same mode. The following gives the modified MILP formulation:

Minimize

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

subject to

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

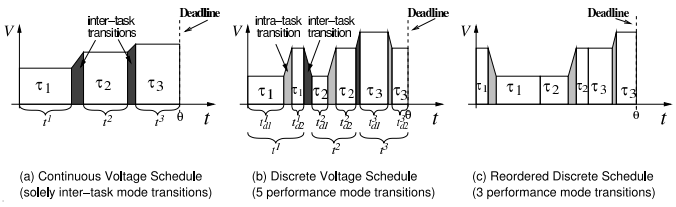


Fig. 5. VS heuristic: mode reordering

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

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

$$D_k + \sum_{s\in\mathcal{M}}\sum_{m\in\mathcal{M}} t_{k,s,m} + \delta_k + \delta_{pl,l} \le D_l \quad \forall (k,l)\in\mathcal{E}, (pl,l)\in\mathcal{E}^{\bullet} \quad (24)$$

$$c_{k,s,i} = t_{k,s,i}\cdot f_i \qquad s\in\mathcal{M},\ i\in\mathcal{M},\ c_{k,s,i}\in\mathbb{N} \quad (25)$$

$$\sum_{s\in\mathcal{M}}\sum_{i\in\mathcal{M}} c_{k,s,i} = NC_k \quad (26)$$

In order to capture the energy overheads in the objective function (Eq. (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 overheads are precomputed based on the available modes (voltage pairs) and Eq. (4). The overall energy overhead is given by all intratask and intertask transitions. The intratask and intertask delay overheads, given in Eq. (21) and (22), are calculated based on a delay penalty matrix $DP_{i,j}$, which, similarly to the energy penalty matrix, can be precomputed based on the available modes and Eq. (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 $1,...,|\mathcal{M}|-1$, i,j in $1,...,m$). These are used in order to capture the intratask overheads, as in Eq. (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 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 Eq. (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. Eq. (25) and (26) are a reformulation of Eq. (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. 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 is used by a subtask. The complete MILP model can be found in [34].

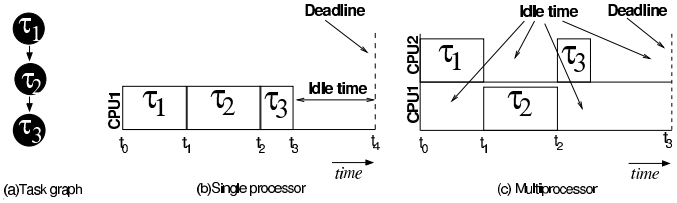


Fig. 6. Schedules with idle times

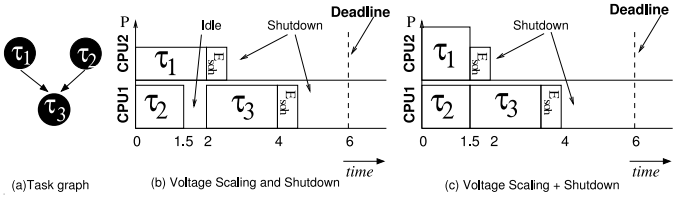


Fig. 7. Voltage Selection with Shutdown

D. Discrete Voltage Selection Heuristic

As shown earlier, discrete voltage selection is NP-hard. Thus, solving it using the presented MILP formulation for large instances is time consuming. We propose a heuristic to effectively solve the discrete voltage selection problem. The main idea behind this heuristic is to perform a continuous voltage selection (as outlined in Section VI). As a result of this calculation, for each task, a continuous voltage pair $(V_{dd_{con}}, V_{bs_{con}})$, as well as the corresponding frequency f_{con} will be determined. Using the approach introduced in [1], for each task the two surrounding discrete performance modes are chosen such that $f_{d1} < f_{con} < f_{d2}$. That is, the execution of a task is split into two regions with t_{d1} and t_{d2} being the execution times in the mode with f_{d1} and f_{d2}, respectively. Fig. 5(a) and 5(b) illustrate this transformation for an application with three tasks. In the continuous scaling case, Fig. 5(a), each task executes at a single voltage level, 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_i for the mode change has to be considered as well, i.e., $t^i = t^i_{d1} + t^i_{d2} + \delta_i$, where t^i is the execution time with continuous voltage setting of the task τ_i. In general, executing tasks in two performance modes, determined as above, leads to close to optimal discrete voltage selection. Having determined the discrete performance mode settings, the inter-task transition overheads are reduced by reordering the mode sequence of each task. We reorder the modes in a greedy manner, such that the inter-task overhead between consecutive tasks is minimized. This is outlined in Fig. 5(c). While this reordering technique is optimal for processors 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.

VIII. Voltage Selection with Processor Shutdown

In this section we discuss the integration of two system level energy minimization techniques: voltage selection and processor shutdown. Voltage selection is effective in minimizing the active energy consumption (the energy consumed while executing a certain task). However, specially in multiprocessor environments, processors alternate between active and idle periods. During idle times, a certain amount of energy, proportional to the length of the idle period is consumed. A solution for saving this energy is to shutdown the processor. The transition to the shutdown state and from shutdown back to operation implies a time and an energy overhead.

Idle times may be present due to multiple reasons, even after performing voltage selection. Consider, for example, the three tasks in Fig. 6(a). If the application runs on a single processor system at the lowest speed, it still finishes before the deadline, as depicted in Fig. 6(b). In the idle interval between the finishing time and the deadline, the processor consumes energy. In this situation, we could shut down the processor and thus save energy. In the case of a single processor system with tasks that do not have arbitrary arrival times, deciding weather or not to shutdown and for how long is relatively easy. In [16], the notion of threshold time interval is defined as the minimul length of an idle period that would provide energy savings by shuting down. A shutdown is decided if the idle interval available is larger than the threshold time.

Imagine now a more complex case, when the application runs on two processors, as in Fig. 6(c). Due to dependencies between tasks that are mapped on different processors, there is a certain amount of slack that cannot be exploited by voltage selection. For example, task τ_2 can start only after task τ_1 has finished. Consequently, there is an idle interval on CPU_1 from time 0, until the start of τ_2. Deciding in this case weather or not to shutdown is a complex problem that will be addressed in the following section.

Even though voltage selection aims at optimizing the active energy, while processor shutdown minimizes the energy consumed during idle periods, these two techniques are not orthgonal. Let us consider an application consisting of 3 tasks, τ_1, τ_2 and τ_3, as in Fig. 7(a). The tasks are mapped on two processors CPU_1 and CPU_2. The resulting schedule, after performing voltage selection is depicted in Fig. 7(b), with all the 3 tasks running at the lowest speeds. Task τ_1 is running for $2ms$ with $200mW$, while τ_2 and τ_3 run at $400mW$ for $1.5ms$ and respectively $2ms$. A brief analysis of the idle times present after voltage selection on both processors, allows us to further reduce the energy consumption by shutting down CPU_1 after the execution of τ_1 and of CPU_2 after τ_3. The energy overhead for shutdown is $75\mu J$ on CPU_1 and $125\mu J$ on CPU_2. We notice the idle interval of $0.5ms$ on CPU_2, between the executions of τ_2 and τ_3. The idle power on CPU_2 is $250mW$, resulting in an energy consumption of $125\mu J$. Please note that the energy consumed during this idle period equals the energy overhead of a shutdown, so it would not pay off to shutdown after τ_2. However, let us consider the possibility of running τ_1 faster, such that it finishes in $1.5ms$. The power consumption that corresponds to this frequency is $300mW$. This slight increase on CPU_1 is compensated by the fact that we can now execute task τ_3 immediately after τ_2, use one shutdown operation to exploit all the idle time on CPU_2 and thus save $125\mu J$.

A. Processor Shutdown: Problem Complexity

The shutdown problem without voltage selection (SNVS) is formulated as follows:

Consider a set of tasks with precedence constraints $\mathcal{T} = \{\tau_i\}$ that have been mapped and scheduled on a set of processors. Each processor operates at a given fixed frequency. For each task τ_i, its deadline dl_i and number of clock cycles to be executed NC_i are given. The start time of each task is variable (with the constraints imposed by the precedences in the scheduled task graph). When a processor is idle, an amount of energy proportional to the length of the idle interval is consumed. In order to save energy, during such an idle interval the particular processor can be shut down. A shutdown operation comes with a fixed time and energy penalty. Our goal is to minimize the energy consumed by the system while the processors are idle. This translates into spending as most as possible of the idle time in the shutdown state. In order to be energy efficient, the best solution will assign the task start times such that idle times are grouped together in big intervals that can be covered with few shutdown operations.

Theorem 2: The shutdown problem (SNVS) is NP-complete.

The proof is given in [34]. It is based on the fact that the multiple choice continuous knapsack problem can be reduced to the SNVS problem. If the simple shutdown problem without performing voltage selection is NP complete, then the combined voltage selection problem with shutdown (even in the case with continuous voltages) is NP complete as well.

B. Continuous Voltage Selection with Processor Shutdown (CVSSH)

In this section we present an exact integer nonlinear formulation as well as a polynomial time heuristic for the voltage selection with processor shutdown [3]. The following gives the modified nonlinear programming formulation (CVSSH):

Minimize

$$\underbrace{\sum_{k=1}^{|\mathcal{T}|} NC_k \cdot C_{eff_k} \cdot V_{dd_k}^2}_{E_{dyn}} + \underbrace{\sum_{k=1}^{|\mathcal{T}|} L_g \cdot (K_3 \cdot V_{dd_k} \cdot e^{K_4 \cdot V_{dd_k}} \cdot e^{K_5 \cdot V_{bs_k}} + I_{Ju} \cdot |V_{bs_k}|) \cdot t_k}_{E_{leak}} + \underbrace{\sum_{k=1}^{|\mathcal{T}|} xi_k \cdot t_{idle_k} \cdot P_{idle_k} + xs_k \cdot (E_{soh_k} + t_{off_k} \cdot P_{off_k})}_{E_{idle}+E_{off}} \quad (27)$$

subject to

$$t_k = NC_k \cdot \frac{(K_6 \cdot L_d \cdot V_{dd_k})}{((1+K_1) \cdot V_{dd_k} + K_2 \cdot V_{bs_k} - V_{th_1})^{\alpha}} \quad (28)$$

[3]For simplicity of the presentation, we omit here the consideration of voltage transition overheads. Nevertheless, these overheads can be easily included, as shown in section VI-B

$$\begin{aligned}
D_k + t_k &\leq D_l \quad \forall (k,l) \in \mathcal{E} - \mathcal{E}^{\bullet} && (29)\\
D_k + t_k + xi_k \cdot t_{idle_k} &= D_l \quad \forall (k,l) \in \mathcal{E}^{\bullet} && (30)\\
D_k + t_k + xs_k \cdot T_{soh_k} + t_{off_k} &= D_l \quad \forall (k,l) \in \mathcal{E}^{\bullet} && (31)\\
xi_k + xs_k &= 1 \quad \forall \tau_k && (32)\\
D_k + t_k &\leq dl_k \quad \forall\ \tau_k \text{ with dl} && (33)\\
D_k &\geq 0 && (34)\\
xi_k, xs_k &\in \{0,1\} && (35)
\end{aligned}$$

$$V_{dd_{min}} \leq V_{dd_k} \leq V_{dd_{max}} \quad \text{and} \quad V_{bs_{min}} \leq V_{bs_k} \leq V_{bs_{max}} \quad (36)$$

There are two noticeable differences between this formulation and the one in section VI-A: the inclusion in the objective (Eq. 27) of the energy spent during idle and shutdown intervals and Eq. 31 and 30 introduced in order to account for the idle and off times. P_{idle_k}, P_{off_k}, E_{soh_k} and T_{soh_k} are constants for each task τ_k and capture the power consumed by the processor on which τ_k is mapped, during idle and shutdown time intervals and respectively the energy and the time overhead associated to a shutdown operation. Please note the usage in Eq. 27, 30 and 31 of binary variables xi_k and xs_k, associated to each task, with the following semantics: if task τ_k is followed by a shutdown, then $xs_k = 1$ and $xi_k = 0$, otherwise $xi_k = 1$ and $xs_k = 0$. In case of a shutdown, t_{off_k} captures the amount of time the processor is off. If there is no shutdown after the execution of τ_k, t_{idle_k} captures the amount of idle time (t_{idle_k} is 0 if the next task starts immediately after τ_k).

The binary variables xi_k and xs_k change the complexity of this nonlinear programming formulation, compared to the ones presented in sections VI-A and VI-B. While the problems presented there are convex nonlinear, the CVSSH problem is integer nonlinear. Indeed, as shown in the previous section, the voltage selection with shutdown problem is NP complete, even in the case when continuous voltage selection is used. Therefore, in the following, we propose a heuristic to efficiently solve the problem.

Let us consider particular instances of the CVSSH problem, where xi_k and xs_k are given constants for each task τ_k. We denote this simplified problem CVSI. Such a particular instance can be solved in polynomial time and computes the optimal voltages for a system in which we know the position of the shutdown operations. For example, if $xi_k = 1$, for all the tasks τ_k, CVSI computes the task voltages such that the energy is minimized, taking into account the idle energy, without performing any shutdown. Running CVSI for all possible combinations for xs_k and xi_k and selecting the one with the minimum energy, provides the optimal solution for the voltage selection with shutdown problem. This is, practically, not possible, of course. We will present in the following a heuristic that solves the CVSSH problem in polynomial time. The pseudocode of the heuristic is given in Fig. 8. The algorithm takes as input the mapped and scheduled task graph with each task characterized as in section V. It returns, the supply and body bias voltage for each task as well as the position and length of each shutdown operation and idle time.

As a first step (line 02), we perform voltage selection, using the CVSI nonlinear formulation. This will optimize the active and idle energy, without performing any shutdown operation

```
Algorithm: CONT_VS_SHUT_HEU
Input: - Mapped and scheduled task graph
       - For each task: NC_k, C_eff_k, dl_k
Output:- V_dd_k, V_bs_k, xs_k, xi_k, tof f_k, tidle_k
01: for all τ_k xs_k = 0, xi_k = 1
02: E_current=call CVSI
03: while(1) {
04:  for all τ_k EFT_k=earliest_start_time(τ_k)
05:  for all τ_k LST_k=latest_start_time(τ_k)
06:  for all (k,l) ∈ E• t_idle_k = LST_l − EFT_k
07:  if ∀τ_k t_idle_k · P_idle_k ≤ E_soh_k break
08:  *select τ_k with t_idle_k · P_idle_k = max{t_idle_l · P_idle_l |τ_l ∈ T}
09:  set xs_k = 1, xi_k = 0
10:  E_current=call CVSI
11: }
12: while(1) {
13:  for all τ_k EFT_k=earliest_start_time(τ_k)
14:  for all τ_k LST_k=latest_start_time(τ_k)
15:  for all (k,l),(l,m) ∈ E• t_idle_k,l,m = LST_m − t_l − EFT_k
16:  if ∀(k,l),(l,m) ∈ E•, t_idle_k,l,m · P_idle ≤ E_soh_k break
17:  *select set σ_k,l,m with
       t_idle_k,l,m · P_idle_k = max{t_idle_h,i,j · P_idle_h |(h,i),(i, j) ∈ E•}
18:  set xs_k = 1, xi_k = 0, xs_l = 0, xi_l = 1
19:  E_1=call CVSI
20:  set xs_k = 0, xi_k = 1, xs_l = 1, xi_l = 0
21:  E_2=call CVSI
22:  *set (xs_k = 1, xs_l = 0) if E_1 > E_current & E_1 > E_2
23:  *set (xs_k = 0, xs_l = 1) if E_2 > E_current & E_2 > E_1
24:  *set (xs_k = 0, xs_l = 0) if E_1 < E_current & E_2 < E_current
25:  E_current = min{E_current, E_1, E_2}
26: }
27: return (V_dd_k, V_bs_k, xs_k, xi_k, t_of f_k, tidle_k)
```

Fig. 8. Voltage Selection with Shutdown Heuristic

($xs_k = 0$ and $xi_k = 1$).

In a second step, (lines 03-11), the idle intervals are inspected one by one, and, if an interval is large enough (line 08) a shutdown is introduced. In more detail, we find iteratively the idle time with the highest energy that is large enough to allow a shutdown. For this purpose, we compute, for each task τ_k, the earliest finishing time EFT_k and the latest start time LST_k (line 04-05), assuming that each task is running at a fixed speed using the voltages computed by CVSI at line 02 or in the previous iteration at line 10. We select for shutdown the idle time that consumes the most energy (line 08-09). We set the corresponding binary variables $xs_k = 1$ and $xi_k = 0$ in order to schedule a shutdown after the task τ_k. Then we run CVSI with the updated values for xi and xs (line 10). At each new iteration the global energy consumption is improved.

When the algorithm exits the loop from lines 03-11, there is no idle interval that is large enough to produce energy savings by a shutdown (line 07). However, in principle, there are two ways to further reduce the consumed energy:

1) Increase the voltages of some tasks such that the idle intervals following them become longer and, thus, can be exploited by shutdowns.

2) Increase the voltages of some tasks such that several idle intervals can be merged and exploited by a single shutdown.

The first alternative can be excluded based on a simple reasoning. Let us assume that we have a task τ_k that runs in mode m_1 and consumes a certain amount energy E_k^1. Task τ_k is followed by an idle interval of length $t_{idle_k}^1$, that is too small to provide savings via shutdown:$t_{idle_k}^1 \cdot P_{idle_k} < E_{soh_k}$. The total energy consumed in this case is $E_k^1 + t_{idle_k}^1 \cdot P_{idle_k}$. Consider that we increase the speed of τ_k by running it with execution mode m_2 instead of m_1. In this case τ_k will consume E_k^2 ($E_k^2 > E_k^1$) and the idle interval becomes long enough to make a shutdown operation efficient. As a result the total energy is $E_k^2 + E_{soh_k}$. Since $E_k^2 > E_k^1$ and $E_{soh_k} > t_{idle_k}^1 \cdot P_{idle_k}$, the energy of the system obtained by running τ_k in execution mode m_2 with a shutdown during the idle time is actually higher than the energy of the system obtained by running τ_k in execution mode m_1 without a shutdown. As a conclusion, increasing the speed of a task such that an idle interval becomes large enough for a shutdown does not provide any energy savings.

The second alternative is illustrated in Fig. 7. The energy is reduced by speeding up certain tasks in order to create the possibility of merging several small idle intervals. In this way, the resulting idle interval can be exploited by a single shutdown operation. This alternative is explored as the third step of our heuristic (lines 12-26). We inspect all the groups of three consecutive tasks mapped on the same processor, τ_k, τ_l and τ_m with $(k,l),(l,m) \in \mathcal{E}^\bullet$ and explore the savings achievable by merging t_{idle_k} and t_{idle_l}. More exactly, for all sets of three tasks $\sigma_{k,l,m} = \{(\tau_k,\tau_l,\tau_m)|(k,l),(l,m) \in \mathcal{E}^\bullet\}$, we compute the maximum set idle time $t_{idle_{k,l,m}}$ as the difference between the latest start time of task τ_m, the execution time of τ_l and the earliest finishing time of τ_k (line 15). We select the set $\sigma_{k,l,m}$ with the highest energy (line 17). For this set, there are two candidate locations of the shutdown operation: after the execution of τ_k or after the execution of τ_l. Our algorithm explores both possibilities (lines 18-21). Using CVSI, we first compute the energy considering the showdown after τ_k (E_1) and secondly after τ_l (E_2). If both E_1 and E_2 are higher then the energy obtained without a shutdown after τ_k and τ_l, no shutdown is scheduled during this iteration (line 24). Otherwise, the algorithm schedules a shutdown after τ_k or after τ_l (lines 22-23). The global energy is improved at each iteration (line 25). The loop exits when no idle time corresponding to a set is large enough to produce savings via shutdown (line 16).

This heuristic relies on a continuous formulation for the computation of the task voltages. We use the heuristic presented in section VII-D in order to translate the computed voltage levels into the discrete ones available on the processors.

IX. Combined Voltage Selection for Processors and Communication Links

In this section, we consider the supply and body bias voltage selection problem for processors and communication links. We introduce a set of communication models for energy and delay estimation. We study two different bus implementations and show the implication of the bus implementation type on the voltage selection strategy. We introduce a nonlinear model of the continuous voltage selection problem, which is optimally solvable in polynomial time, while for the discrete voltage selection case we use a heuristic similar to the one presented in section VII-D. For simplicity of the explanation, we have

not considered the processor shutdown during the formulation of the optimization problems in this section, however, the extension is straightforward.

A. Voltage Selection on Repeater-Based Buses

Consider an architecture consisting of two voltage-scalable processing elements (CPU1 and CPU2) that communicate via a repeater-based, shared bus (CL1), which also allows voltage selection. CPU1 executes task τ_1 and CPU2 runs τ_2. Task τ_2 can only start after receiving data from τ_1, and it has to finish execution before a deadline of $2ms$. Fig. 9(a) shows the schedule for this system, considering an execution at the nominal voltage settings (highest supply voltage and body bias voltage). The diagram shows the energy dissipation (dynamic

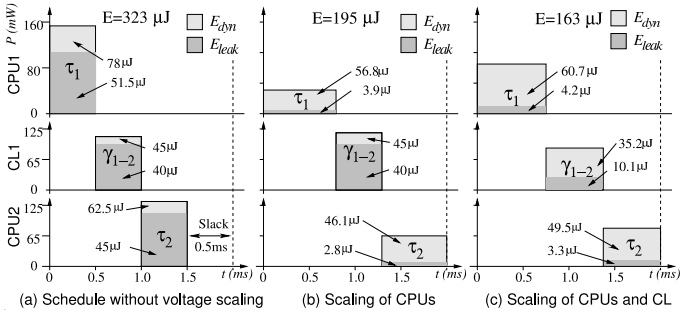


Fig. 9. Voltage selection on a repeater-based bus

and leakage) of the individual components. For clarity we assume in this example that the processors as well as the repeaters of the bus have the same nominal voltage values ($V_{dd} = 1.8V$ and $V_{bs} = 0V$). Furthermore, we assume that the supply voltages and the body bias voltages of all components can be varied continuously in the ranges $[0.6, 1.8]V$ and $[-1, 0]V$, respectively. Given the power consumptions at the nominal voltages, we can compute a total energy consumption of the tasks and communication in the initial schedule as $(156+103)mW \cdot 0.5ms + (90+80)mW \cdot 0.5ms + (125+90)mW \cdot 0.5ms = 323\mu J$. As can be observed, at the nominal voltages the system over-performs, leading to a slack of $0.5ms$.

We can exploit this slack by scaling the voltages of the processing elements. Using the technique described in section VI, the resulting voltages for tasks τ_1 and τ_2 are $(1.43V, -0.42V)$ and $(1.54V, -0.49V)$, respectively. The corresponding, voltage scaled schedule is shown in Fig. 9(b). The dynamic and leakage power consumptions of the tasks are reduced to $(72mW, 5mW)$ and $(65mW, 4mW)$; however, the execution times have increased to $0.79ms$ and $0.71ms$. With these settings, the system dissipates $195\mu J$, a reduction by 39% compared to the energy at nominal voltages.

To demonstrate the importance of combined voltage selection of the processors and the repeater-based bus, we have produced the schedule in Fig. 9(c). The optimal voltage settings can be calculated as $(1.48V, -0.42V)$ for CPU1, $(1.77V, -0.61V)$ for CPU2, and $(1.59V, -0.50V)$ for the bus repeaters. Correspondingly the power dissipations are $(81mW, 5.6mW)$, $(73.8mW, 4.9mW)$ and $(55.8mW, 16mW)$ thereby, reducing the overall system energy dissipation to $163\mu J$. This is a reduction of 49% compared to the nominal energy consumption, which is 10% more than in the case when only the PEs are voltage scaled.

B. Voltage Swing Selection on Fat Wire Buses

In this example, we illustrate the influence that a dynamic variation of the voltage swing (the voltage on the wire) has on the energy efficiency of the bus. Fig. 10 shows the total power consumption of a fat wire bus (including drivers and receivers), depending on the voltage swing at which data is sent. These plots have been generated via SPICE simulations using the Berkeley predictive $70nm$ CMOS technology library. The two plots show the total power consumption on the bus for two different voltage settings of the bus drivers and receivers. For example, if the driver connected to CPU1 and the receiver at CPU2 operate at $1.0V$, the lowest bus power dissipation ($0.55mW$) is achieved by a voltage swing of $0.14V$. Let us

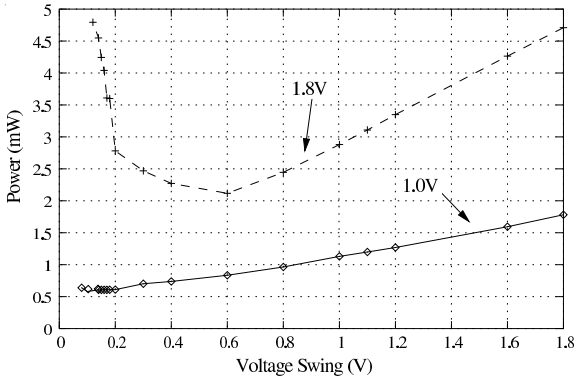


Fig. 10. Optimum swing on a fat wire bus

assume that the voltages of the driver and receiver are changed during run-time to $1.8V$ due to voltage selection. The bus power/voltage swing relation for this situation is indicated by the dashed line. As we can observe, by keeping the voltage swing at $0.14V$, the power dissipation on the bus will be $4.5mW$. However, inspecting the plot reveals that it is possible to reduce the bus power dissipation by changing the voltage swing from $0.14V$ to $0.6V$. At this voltage swing, the bus dissipates a power of $2.2mW$, i.e., a 51% reduction can be achieved by changing the voltage swing.

Now assume that the driver and receiver voltages are changed back from $1.8V$ to $1.0V$. Keeping the swing at $0.6V$ results in a power of $0.83mW$, which is, compared to the optimal $0.55mW$ at $0.14V$, 33% higher than necessary.

C. Communication Models

We consider a bus-based communication system as in Fig. 11. Whenever the processor CPU_1 sends data to CPU_2 over the bus, V_{dd_1} is converted to the bus voltage V_{dd_3} by the bus adapter of CPU_1. At the destination processor CPU_2, V_{dd_3} is converted to V_{dd_2}. Each voltage conversion in the bus adapter requires an energy overhead, which is:

$$E_{adapter} = C_{adapter} \cdot (V_{dd_{CPU}} - V_{dd_{bus}})^2 \qquad (37)$$

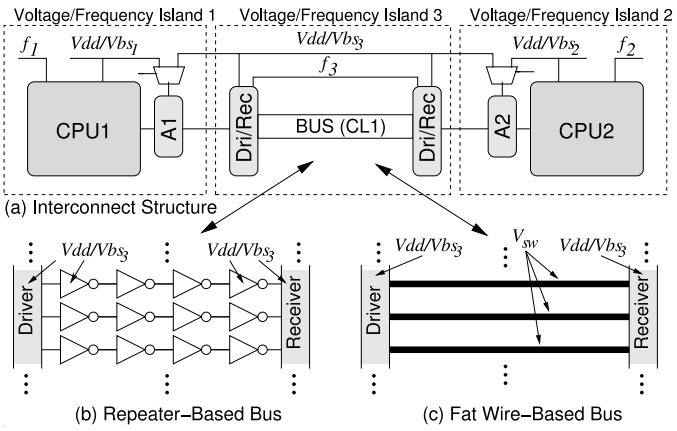


Fig. 11. Interconnect structures

Thus, the total energy consumed when communicating between two processors CPU_1 and CPU_2 over the bus is:

$$E_{comm} = E_{adapter_1} + E_{bus} + E_{adapter_2} \tag{38}$$

Feature size scaling in deep-submicron circuits is responsible for an increasing wire delay of the global interconnects. This is mainly due to higher wire resistances caused by a shrinking cross-sectional area. Two approaches to cope with this problem have been proposed: (a) the usage of repeaters [19], [20] and (b) the usage of fat wires [17], [18]. The bus energy E_{bus} in Eq. (38) depends on which of these two approaches is used.

1) Repeater-Based Bus: The wire delay depends quadratically on the wire length, which can be approximated using an *RC* model. In order to reduce this quadratic dependency, it is possible to break the wire into smaller segments by inserting repeaters. The authors in [18] estimate an increasing number of repeaters with technology scaling down. For instance, up to 138 repeaters are used in $50nm$ technology for a corner-to-corner wire with a die size of $750mm^2$. Repeaters are implemented as simple CMOS inverter circuits (Fig. 11(b)). In accordance, the power dissipated by a bus implemented with repeaters is given by,

$$P_{rep} = N \cdot (\underbrace{s_\tau \cdot C_{rep} \cdot V_{dd}^2 \cdot f}_{P_{dyn}} + \underbrace{V_{dd} \cdot K_3 \cdot e^{K_4 \cdot V_{dd}} \cdot e^{K_5 \cdot V_{bs}} + |V_{bs}| \cdot I_{Ju}}_{P_{leak}}) \tag{39}$$

where N is the number of repeaters, s_τ is the average switching activity caused by communication task $\tau \in \mathcal{K}$, C_{rep} is the load capacity of a repeater (the sum of the output capacity of a repeater C_d, the wire capacity C_w, and the input capacity of the next repeater C_g), and V_{dd}, V_{bs}, and f are the supply voltage, body bias voltage, and the frequency at which the repeaters operate. Further, the constants K_3, K_4, K_5, and I_{Ju} depend on the repeater circuits (see section III).

The bus speed is constrained by the repeater frequency. Since repeaters are implemented as CMOS inverters, we use Eq. (3) to approximate the operational frequency f of the bus. The execution time of a communication $\tau \in \mathcal{K}$ is given by,

$$t = \left\lceil \frac{NB_\tau}{W_{bus}} \right\rceil \cdot \frac{1}{f} \tag{40}$$

where NB_τ denotes the number of bits to be transmitted by communication τ and W_{bus} is the width of the bus (i.e. the number of bits transmitted with each clock cycle). Accordingly to Eq. (39) and (40), the bus energy dissipation is given by $E_{bus} = P_{rep} \cdot t$. Scaling the supply and body bias voltage of the repeaters requires also an overhead in terms of energy and time, similar to the overheads required by processor voltage selection (see Eq. (4) and (5)).

2) Fat Wire-Based Bus: Another approach for reducing the wire delay is to increase the physical dimensions of the wire, instead of scaling them down with technology. The usage of "fat" wires, on the top metal layer, has been proposed in [17]. The main advantage of such wires is their low resistance. Provided that $L \cdot R_w / Z_0 < 2ln2$ (L is the wire length, R_w is the wire resistance per unit length and Z_0 its characteristic impedance), they exhibit a transmission line behavior, as opposed to the *RC* behavior in the repeater-based architecture. Using fat wires, the transmission speed approaches the physical limits (the speed of light in the particular dielectric). However, only a limited wire length can be accomplished with the available width of the top metal layer. For example, for a $4mm$ long wire in $180nm$ technology, the authors in [37] obtained a fat wire width of $2\mu m$ on the top metal layer.

The dynamic power consumption of a fat wire-based bus is mainly due to its large line capacitance. This capacitance is driven by a driver, with the dynamic power consumption:

$$P_{dri_{dyn}} = s_\tau \cdot f \cdot (C_{dri} + C_w) \cdot V_{dd}^2 \tag{41}$$

where s_τ is the switching activity caused by communication task $\tau \in \mathcal{K}$, f is the bus frequency, and C_{dri} and C_w represent the capacitance of the driver and the wire, respectively.

One way to limit the dynamic power is to transmit data at a lower voltage swing, V_{sw}, instead of using the higher bus voltage V_{dd}. Correspondingly, the dynamic power consumed by the driver is given by:

$$P_{dri_{dyn}} = \begin{cases} s_\tau \cdot f \cdot (C_{dri} + C_w) \cdot V_{dd} \cdot V_{sw} & \text{if } V_{sw} \text{ is generated on chip} \\ s_\tau \cdot f \cdot (C_{dri} + C_w) \cdot V_{sw}^2 & \text{otherwise} \end{cases} \tag{42}$$

The driver dissipates a non-negligible leakage power

$$P_{dri_{leak}} = L_g \cdot (V_{dd} \cdot K_3 \cdot e^{K_4 \cdot V_{dd}} \cdot e^{K_5 \cdot V_{bs}} + |V_{bs}| \cdot I_{Ju}) \tag{43}$$

Since the lower swing corresponds to lower signal values, a receiver has to restore the "original" signal. This requires an amplification, for which a dynamic and a leakage power consumption can be calculated as:

$$P_{rec_{dyn}} = s_\tau \cdot f \cdot C_{rec} \cdot V_{dd}^2 \tag{44}$$

$$P_{rec_{leak}} = L_g \cdot (V_{dd} \cdot K_3 \cdot e^{K_4 \cdot V_{dd}} \cdot e^{K_L \cdot (V_{dd}/2 - V_{sw}/2)} \cdot e^{K_5 \cdot V_{bs}} + |V_{bs}| \cdot I_{Ju}) \tag{45}$$

Please note that the leakage power exponentially depends on the difference between the bus voltage V_{dd} and the voltage swing V_{sw} (K_L is a technology dependent parameter), i.e., a lower voltage swing results in a higher static energy (while the dynamic power is reduced, Eq. 42). In order to find the most efficient solution we need to find an appropriate voltage swing that minimizes the total bus power $P_{bus} = P_{dri_{dyn}} + P_{dri_{leak}} + P_{rec_{dyn}} + P_{rec_{leak}}$. Using the optimal voltage

swing can significantly reduce the power consumption of the bus [37], [17].

The speed at which the data can be transmitted over the fat wires can be considered to be independent of the voltage swing V_{sw}. Yet, the bus driver and receiver circuits introduce a delay that depends on the voltages V_{dd} and V_{bs}. This delay d and the corresponding operational frequency can be calculated according to Eq. (3). In order to lower the power dissipation of the drivers and receivers, it is possible to reduce V_{dd} and/or to increase V_{bs}, which, in turn, necessitates the reduction of the bus speed. However, it is important to note that the optimal voltage swing depends on the V_{dd} and V_{bs} settings of the drivers and receivers (see Fig. 10). Since these settings are dynamically changed during run-time via voltage selection, the value of the optimal voltage swing changes as well during run-time, and has to be adapted accordingly.

In addition to the transition overheads in terms of energy and time, which are required when scaling the voltages of the drivers and receivers (see Eq. (4) and (5)), the dynamic scaling of the voltage swing necessitates additional overheads. For a transition from V_{sw_j} to V_{sw_k} these overheads in energy and time are given by,

$$\varepsilon_{k,j} = C_{wr} \cdot (V_{sw_k} - V_{sw_j})^2 \quad \text{and} \quad \delta_{k,j} = p_{Vsw} \cdot |V_{sw_k} - V_{sw_j}| \tag{46}$$

where C_{wr} is the wire power rail capacitance and p_{Vsw} is the time/voltage slope.

D. Problem Formulation

We assume that all computation tasks and communications have been mapped and scheduled onto the target architecture. For each computation task $\tau_i \in \Pi$ its deadline dl_i, its worst-case 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 voltage selection problem), or within a set of discrete voltages pairs $m_z = \{(V_{dd_z}, V_{bs_z})\}$ (for the discrete voltage selection problem). A transition between two different performance modes on a processor requires a time and an energy overhead.

For each communication task $\tau_k \in \mathcal{K}$, the number of bytes NB_k is given. Depending on the employed bus implementation style, either using repeaters or fat wires, we have to distinguish between two subproblems:

Repeater Implementation: The communication speed as well as the communication power on bus architectures implemented through repeaters depend on the supply voltage and body bias voltage. Similar to processing elements, these voltages can be varied within a continuous range, or within a set of discrete voltage pairs $m_z = \{(V_{dd_z}, V_{bs_z})\}$, and transitions between different bus performance modes require an energy and time overhead. Furthermore, an energy overhead is required to adapt the bus voltage to the processor voltage.

Fat Wire Implementation: If communication is performed over fat wires, it is necessary to dynamically adapt the voltage swing at which data is transfered. Furthermore, in order to reduce the power dissipated by the bus drivers and receivers, it is possible to dynamically scale the supply and body bias voltage of these components. While the voltage swing can be scaled without an influence on the bus speed, the operational speed of the bus drivers and receivers is affected through voltage selection, i.e., the bus performance has to be adjusted in accordance to the driver/receiver speed. In the case of continuous voltage selection, the value for the voltage swing, the supply voltage, and the body bias voltage can be changed within a continuous range. On the other hand, for the discrete voltage selection case, the components operate across sets of discrete voltages, referred to as modes. For the voltage swing this set is $n_z = \{V_{sw_z}\}$ and for the bus drivers and receiver the set is $m_z = \{(V_{dd_z}, V_{bs_z})\}$. Of course, changing the voltage swing value as well as the supply and body bias voltages requires an energy and time overhead. □

Our overall goal is to find mode assignments for each processing and communication task, such that the individual task deadlines are satisfied and the total energy consumption, including overheads, is minimal.

E. Voltage Selection with Processors and Communication Links

We introduce a nonlinear programming model of the continuous voltage selection problem formulated in section IX-D which is optimally solvable in polynomial time, as follows:

Minimize

$$\underbrace{\sum_{k}^{|\Pi|} E_{dyn_k} + E_{leak_k}}_{\text{computation}} + \underbrace{\sum_{k}^{|\mathcal{K}|} E_{dyn_k} + E_{leak_k}}_{\text{communication}} + \underbrace{\sum_{(k,j)\in\mathcal{E}^\bullet} \varepsilon_{k,j}}_{\text{overhead}} \tag{47}$$

subject to

$$t_k = \begin{cases} NC_k \cdot \frac{(K_6 \cdot L_d \cdot V_{dd_k})}{((1+K_1)\cdot V_{dd_k} + K_2 \cdot V_{bs_k} - V_{th_1})^\alpha} & \text{if } \tau_k \in \Pi \\ \left\lceil \frac{NB_k}{W_{bus}} \right\rceil \cdot \frac{(K_6 \cdot L_d \cdot V_{dd_k})}{((1+K_1)\cdot V_{dd_k} + K_2 \cdot V_{bs_k} - V_{th_1})^\alpha} & \text{if } \tau_k \in \mathcal{K} \end{cases} \tag{48}$$

$$D_k + t_k \leq D_l \quad \forall (k,l) \in \mathcal{E} \tag{49}$$

$$D_k + t_k + \delta_{k,l} \leq D_l \quad \forall (k,l) \in \mathcal{E}^\bullet \tag{50}$$

$$D_k + t_k \leq dl_k \quad \forall \tau_k \in \Pi \text{ with a deadline} \tag{51}$$

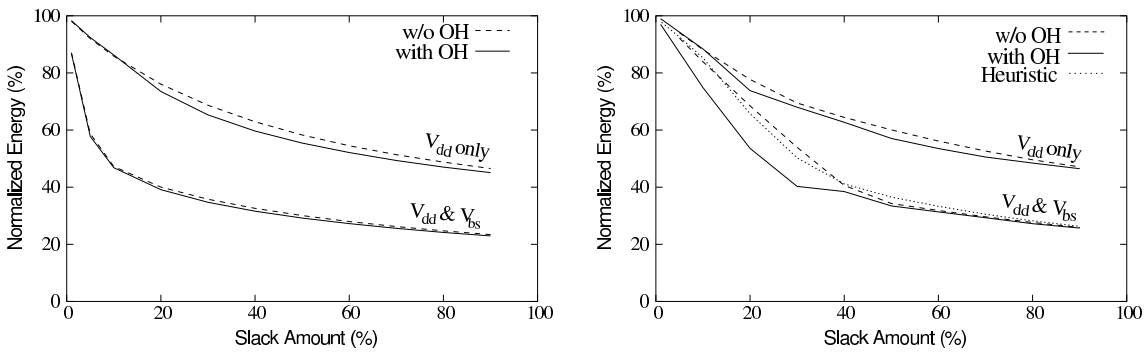
$$D_k \geq 0 \tag{52}$$

$$V_{dd_{min}} \leq V_{dd_k} \leq V_{dd_{max}} \tag{53}$$

$$V_{bs_{min}} \leq V_{bs_k} \leq V_{bs_{max}} \tag{54}$$

$$V_{sw_{min}} \leq V_{sw_k} \leq V_{sw_{max}} \tag{55}$$

The variables that need to be determined are the task and communication execution times t_k, the start times D_k, as well as the voltages V_{dd_k}, V_{bs_k}, and V_{sw_k}. The whole formulation can be explained as follows. The total energy consumption (Eq. (47)), with its three contributors (energy consumption of tasks, communication, and voltage transitions) has to be minimized. For all these energies both their dynamic and active leakage components are considered. The dynamic energy of tasks and communications is given by the following equations



(a) Continuous Voltage Selection

(b) Discrete Voltage Selection

Fig. 12. Optimization Results for Processor DVS & ABB

(derived from the equations discussed in Section III):

$$E_{dyn_k} = \begin{cases} NC_k \cdot s_k \cdot C_{eff_k} \cdot V_{dd_k}^2 & \text{if } \tau_k \in \Pi \\ \sum^N \lceil \frac{NB_k}{W_{bus}} \rceil \cdot s_k \cdot C_{rep} \cdot V_{dd_k}^2 & \text{if } \tau_k \in \mathcal{K} \text{ on repeaters} \\ \lceil \frac{NB_k}{W_{bus}} \rceil \cdot s_k \cdot C_{fat} \cdot V_{dd_k} \cdot V_{sw_k} & \text{if } \tau_k \in \mathcal{K} \text{ on fat wires (intern)} \\ \lceil \frac{NB_k}{W_{bus}} \rceil \cdot s_k \cdot C_{fat} \cdot V_{sw_k}^2 & \text{if } \tau_k \in \mathcal{K} \text{ on fat wires (extern)} \end{cases} \quad (56)$$

where $C_{rep} = C_d + C_w + C_g$ and $C_{fat} = C_{dri} + C_w + C_{rec}$ are the total capacitances that have to be charged by bus implementation either repeater-based or fat wire-based, respectively. Furthermore, in the case of fat wire implementations we have to distinguish between the chip-intern or chip-extern generation of the voltage swing.

The leakage power dissipation of processors and repeater-based buses is:

$$E_{leak_k} = L_g(K_3 \cdot V_{dd_k} \cdot e^{K_4 \cdot V_{dd_k}} \cdot e^{K_5 \cdot V_{bs_k}} + I_{Ju} \cdot |V_{bs_k}|) \cdot t_k \quad (57)$$

For fat wire-based buses we need to additionally account for the leakage in the receiver (see Eq. (43) and (45)), given by,

$$E_{leak_k} = (P_{dri_{leak}} + P_{rec_{leak}}) \cdot t_k \quad (58)$$

The energy overhead due to voltage transitions is given by Eq. (4) and (46).

The constraints are similar to the ones in section VI, expressing the execution order imposed by the scheduling and task graph dependencies, as well as the time constraints.

We use a heuristic similar to the one presented in section VII-D in order to translate the computed continuous voltages into the discrete ones available for the processors and buses.

X. Experimental Results

We have conducted several experiments using numerous generated benchmarks as well as two real-life examples, in order to demonstrate the efficiency of the presented approaches.

A. V_{dd} and V_{bs} Selection on the Processors

The first set of experiments was conducted in order to demonstrate the achievable energy savings when comparing the classic V_{dd} selection with simultaneous V_{dd} and V_{bs} selection. The automatically generated benchmarks consist of 100 task graphs containing between 50 and 150 tasks, which are mapped and scheduled onto architectures composed of 2 to 3 processors (we have considered that all processors are Crusoe TM5600). 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, [2]. For experimental purpose the amount of deadline slack in each benchmark was varied over a range 0 to 90%, using a 10% increment, resulting in 900 performed evaluations. The continuous voltage ranges were set to $0.6V \leq V_{dd} \leq 1.8V$ and $-1V \leq V_{bs} \leq 0$. The values for C_r, C_s, p_{Vdd}, and p_{Vbs} were set to $10\mu F$, $40\mu F$, $100\mu s/V$, and $100\mu s/V$, respectively. Fig. 12(a) shows the outcomes for the continuous voltage selection with and without the consideration of transition overheads. 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 influence of the overhead on the overall energy consumption, we can see that the savings are around 1% for the combined scheme and 2% for the Vdd-only 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 Eq. (4) and Eq. (5)).

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 12(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 (around 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. Compare, for instance, the cases where the combined V_{dd} and V_{bs} selection has been optimized with and without considering the overheads. Between 10% to 40% of slack, the consideration of transition overheads results in solutions with up to 12% higher savings. Of course, with an increasing amount of 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 reduces.

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 40 tasks (with overhead, DOH), since the required optimization times become intractable, as a result of the NP-hardness of the problem (Section VII-A). To overcome this problem we have additionally investigated the voltage selection heuristic proposed in Section VII-D. The results of the heuristic are shown by the dotted line in Fig 12(b) and, as we can be seen, they are close to the optimal (maximum 8% deviation) solution. Moreover, due to its relatively reduced polynomial time complexity, it can be applied to large instances of the problem. At this point it is interesting to note that the optimization times for individual applications with up to 300 tasks using continuous voltage selection were below 1 minute, using the MOSEK optimization software [38] on a 2GHz AMD Athlon PC. Typically, for task graphs with less then 100 tasks, the optimization time is below 15 seconds. The discrete voltage selection without the consideration of the transition overheads, runs between 5 and 20 minutes, for tasks graphs with less then 90 tasks. When considering the overheads during the discrete optimization, an important parameter that affects the optimization time, besides the number of tasks, is the number of execution modes. We were not able to solve optimally task graphs with more then 30 tasks, considering 3 or more execution modes. Even for such a small number of tasks, the optimization time is around 1 hour. The proposed heuristic for discrete voltages, however, has a runtime comparable to the continuous voltage optimization, making it suitable for large applications.

B. Voltage Selection with Processor Shutdown

Using the same setup as in the previous experiments, we have studied the achievable energy savings that can be obtained by using the proposed voltage selection with shutdown, presented in Section VIII-B. We have assumed that the overheads for a shutdown operation are $E_{soh} = 300\mu J$ and $t_{soh} = 1ms$, as in [16]. The results are presented in Fig. 13. On the x axis, we have varied the amount of available deadline slack. We plotted with the continuous line the energy savings achievable by the combined voltage selection and shutdown heuristic presented in section VIII-B, relative to a system that is optimized using solely DVS and ABB, without shutdown. In order the measure

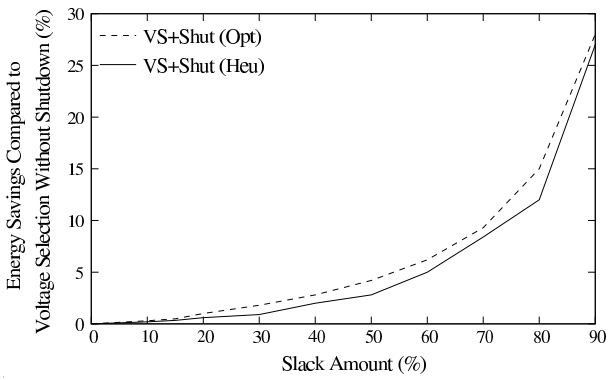


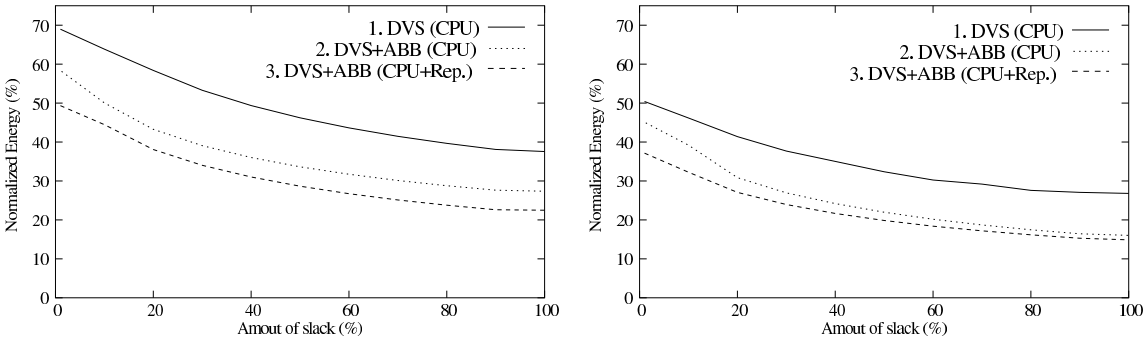
Fig. 13. Voltage Selection with Shutdown

the quality of the heuristic, we have also represented with a dotted line the results obtained by an optimal solution. As we can observe from Fig. 13, if the amount of slack is low, shutting down does not yield additional energy savings. However, the additional benefit of the shutdown is significant for larger amounts of slack. For example, for systems having 30% slack, the additional savings obtained with shutdown, relative to DVS and ABB are only 2%. When the available slack is above 60%, the savings due to shutdown range from 10% to almost 30%. It is interesting to note that the proposed heuristic yields results that are close to the optimal solution.

C. Combined Voltage Selection for Processors and Communication

We have conducted a set of experiments in order to validate the presented techniques for combined processor and bus voltage selection. The automatically generated benchmarks consist of 120 task graphs containing between 50 and 300 tasks, which are mapped and scheduled onto architectures composed of 2 to 5 processors, interconnected via 1 to 4 buses either implemented repeater-based or fat wire-based. The continuous voltage ranges were set to $0.6V \leq V_{dd} \leq 1.8V$ and $-1V \leq V_{bs} \leq 0$, while the discrete voltage levels are $m_z = \{(1.8,0),(1.4,-0.2),(0.8,-0.6),(0.6,-1)\}$. The voltage ranges for repeater-based systems are identical to the possible processor voltage settings. For the fat wire-based buses the continuous voltage swing can be set between 0.2 and $1V$, and for the discrete case it can be adjusted to $m_z = 0.2, 0.3, 0.4, 0.6, 1V$. The amount of deadline slack in each benchmark was varied over a range 0 to 100%, using a 10% increment. Furthermore, the amount of communication within the generated benchmarks was varied between 10 to 50% of the total execution time, with an increment of 10%. Overall, these experiments resulted in 2400 performed evaluations.

The first set of experiments was conducted with the aim to investigate the energy savings that are achievable when dynamically scaling the supply voltage as well as body bias voltage of bus repeaters. The 32bit-wide bus architecture under consideration consisted of 27 repeaters per bit-line of which each has a total length of $27.4mm$. The capacitance of a single wire including the repeaters was estimated as $7.2pF$, using the power optimized data from [39]. Fig. 14(a) shows



(a) Energy dissipation of a reapter–based system

(b) Energy dissipation of a fat wire–based system

Fig. 14. Optimization Results for Different Bus Implementations

the outcomes of three system configurations for different amounts of system slack. All plots have been normalized against the energy dissipation at nominal (highest) voltages. The first plot gives the energy consumption for systems in which the repeaters' voltages are kept fixed, while the supply voltage (but not the body-bias voltage) of the processors is dynamically scaled. The second plot represents a system in which the repeater settings are still kept fixed, while combined V_{dd} and V_{bs} scaling is applied to the processors. The third plot indicates the systems in which the repeater-based bus as well as the processors are scaled by changing V_{dd} and V_{bs}. Please note that Fig. 14(a) gives the energy values for systems with a communication amount of 30%, compared to the total execution time. Inspecting the graphs reveals that the highest energy savings are achieved by considering the combined V_{dd} and V_{bs} continuous voltage selection scheme on the buses as well as on the processors (plot 3). We can also observe that the energy efficiency is increased by approx. 12% if combined voltage selection is applied on the bus (difference between plot 2 and 3). Generally, the combined V_{dd} and V_{bs} scaling yields higher energy saving (around 30%) than the V_{dd}-only scaling (difference between plot 1 and 2). the results for continuous voltage selection, it is interesting to note that the proposed heuristic for discrete voltage selection (Section VII-D) achieves results that are within 4% of the values obtained at continuous voltage levels. It is important to note that the efficiency difference of about 12% on average, between implementations with and without bus voltage selection is preserved also when discrete voltage levels are used.

In the second set of experiments, shown in Fig. 14(b), we investigate the achievable energy savings on a fat wire-based bus system, assuming the same bus-width as in the previous experiment. Since fat wires are considered to be suitable only for short distance connections, we consider a length of $4mm$ with a single line capacitance of $609fF$. Similarly to the previous experiments, the plots 1 and 2 represent systems in which only the processing elements are scaled (V_{dd} only for plot 1 and combined V_{dd} and V_{bs} for plot 2), while the third plot indicates systems in which the processors and buses are voltage scaled in terms of V_{dd}, V_{bs}, and V_{sw}. As expected, the fully voltage scalable systems, achieve the best energy savings, with reductions between 4% to 18% compared to systems with fixed bus voltages. Again, applying the heuristic for discrete voltage selection shows that results comparable to the continuous case (within 4%) can be achieved.

Please note that we do not advocate here repeater-based or fat wire-based approaches and do not try to show that one is better than the other. What we do show is that energy savings can be achieved if voltage selection is applied on the communication links and that the communication energy models are highly dependent on the actual technique used to implement the communication lines. The experiments have also shown that with an increasing amount of communication data, the bus voltage selection approach achieves increasingly higher energy reductions. If, for example, the time spent for communications is around 15% of the total execution time, the energy savings due to bus voltage scaling are around 10%. With communication time around 30%, the energy savings become around 16%.

D. Real-Life Examples

We have conducted experiments on two real-life applications: a GSM voice codec and a generic multimedia system (MMS), that includes a H263 video encoder and decoder and MP3 audio encoder and decoder . Details regarding these applications can be found in [40] and [41]. The GSM voice codec consists of 87 tasks and is considered to run on an architecture composed of 3 processing elements with two voltage modes ($(1.8V, -0.1V)$ and $(1.0V, -0.6)$). At the highest voltage mode, the application reveals a deadline slack close to 10%. Switching overheads are characterized by $C_r = 1\mu F$, $C_s = 4\mu F$, $p_{Vdd} = 10\mu s/V$, and $p_{Vbs} = 10\mu s/V$. Tab. I shows the results in terms of dynamic E_{dyn}, leakage E_{leak}, overhead ε, and total energy E_{active} (Columns 2–5). 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 (10.7%) than the overhead neglecting optimization (8.7%). 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 22%, yet taking the overheads into account

Approach	E_{dyn} (*mJ*)	E_{leak} (*mJ*)	ε (*mJ*)	E_{active} (*mJ*)	*Reduction* (%)
Nominal	1.342	0.620	non	1.962	—
DVDDNOH	1.185	0.560	0.047	1.792	8.7
DVDDOH	1.190	0.560	0.003	1.753	10.7
DNOH	1.253	0.230	0.048	1.531	22.0
DOH	1.255	0.230	0.002	1.487	24.3
Heuristic	1.271	0.250	0.008	1.529	22.1

TABLE I

OPTIMIZATION RESULTS FOR THE GSM CODEC

results in a reduction of 24.3%, solely achieved by decreasing the transition overheads. Compared to the classical voltage selection scheme, the combined selection achieved a further reduction of 14%. The last line shows the results of the proposed heuristic approach. It should be noted that, since the problem is NP hard, such heuristic techniques are needed when dealing with larger cases (increased number of voltage modes and tasks). In the GSM application, although the number of tasks is relatively large, we considered only two voltage modes. Therefore the optimal solutions could be obtained for the DOH problem.

We have performed the same set of experiments on the MMS system consisting of 38 tasks that is considered to run on an architecture composed of 4 processors with four voltage modes ($(1.8V, 0.0V)$, $(1.6V, -0.8)$, $(1.3V, -0.9)$ and $(1.0V, -0.9)$). At the highest voltage mode, the application reveals a deadline slack close to 40%. Tab. II shows the results in terms of dynamic E_{dyn}, leakage E_{leak}, overhead ε, and total E_{active} energy (Columns 2–5). As with the GSM,

Approach	E_{dyn} (*mJ*)	E_{leak} (*mJ*)	ε (*mJ*)	E_{active} (*mJ*)	*Reduction* (%)
Nominal	14.88	12.05	non	26.93	—
DVDDNOH	11.33	9.45	0.68	21.46	20.4
DVDDOH	11.31	9.46	0.0001	20.77	22.9
DNOH	11.40	7.18	0.89	19.47	27.7
DOH	11.41	7.18	0.01	18.60	31.0
Heuristic	11.62	7.30	0.40	19.32	29.3

TABLE II

OPTIMIZATION RESULTS FOR THE MMS SYSTEM

the consideration of overheads achieves higher energy savings (22.9% for the Vdd-only selection and respectively 31.0% for the combined approach) than the overhead neglecting optimization (20.4 and respectively 27.7%). Compared to the classical voltage selection scheme (22.9% savings), the combined selection achieved a further reduction of 8.1%.

We have performed a set of experiments on each of the two real-life applications in order to show the efficiency of the proposed voltage selection with processor shutdown technique. The voltage modes are the same for GSM codec and respectively for the MMS system as the ones used in the previous experiments. The results are presented in tables III and IV. Each line represents a different approach. The first line (Nominal) is the baseline and represents an execution at the highest voltages, without any processor shutdown. The remaining four lines represent the resulting energy consumptions for supply voltage selection without (DVddNoSH) and with shutdown (DVddSH) and respectively the supply and body bias selection without (DVddVbsNoSH) and with shutdown (DVddVbsSH). For each approach we list the active (E_{active}), idle and total energy (E_{idle}) consumption. The overheads for a shutdown operation are estimated in [16] as $E_{soh} = 300\mu J$ and $t_{soh} = 1ms$. If we use these values for the GSM voice codec, we can not perform do any shutdown, due to the little amount of slack available after voltage selection. If we consider lower shutdown overheads ($E_{soh} = 90\mu J$ and $t_{soh} = 0.3ms$), we obtain the results presented in table III. As we can see, even considering a reduced overhead, the energy can be improved via shutdown by only 4%. It is interesting to compare the active and idle energy values resulted after performing voltage selection without and with processor shutdown from the lines 4 and 5 in table III. As we can see, the active energy is slightly increased when we perform the shutdown (from $1.48mJ$ to $1.50mJ$), while the idle energy is reduced (from $0.93mJ$ to $0.70mJ$). This means that a situation similar to the one described in Fig. 7 is encountered during the optimization (the voltages for a task are increased in order to allow the merging of several idle intervals into one big shutdown period). The difference between the total energy (E_{total}) and the sum of active (E_{active}) and idle (E_{idle}) energies represents the energy corresponding to the shutdown overheads plus the low energy consumed in the shutdown state. A simple calculation shows that only one shutdown is perfomed in case of the GSM voice codec.

A similar experiment was performed for the MMS. We have used the shutdown overheads estimated in [16] ($E_{soh} = 300\mu J$ and $t_{soh} = 1ms$). The results are presented in table IV. It is interesting to note that performing shutdown in conjunction with supply voltage selection provides a reduction of 9%, compared to a reduction of 5% obtained by the shutdown with the combined V_{dd} and V_{bs} selection. This is due to the fact that the combined supply and body bias voltage selection exploits more slack than the supply-only voltage selection, thus leaving less idle time for potential shutdown operations. As opposed to the GSM voice codec, the optimization determines 5 shutdowns for the MMS.

The relatively reduced energy savings achievable by shutdown are due to the small amount of static slack available. Exploiting the dynamic slack, resulted online from the tasks that execute less then their worst case number of clock cycles, provides an additional opportunity for shutdowns. This is due to the fact that considering the dynamic slack in addition to the static one, provides a higher chance to find, online, large idle periods that can be exploited for shutdown. We have presented in [30] an online voltage selection technique that can make use of dynamic slack. The technique is based on an offline calculation of look-up tables that are used online for voltage selection. The calculation of the tables is based on the equations presented in this paper. Applied on top of such an approach, a strategy which includes shutdown produces its

entire potential. For example, for the MMS system, in the case that the average execution time of the tasks is half of the worst case, we can achieve a further energy reduction of 60% by using the shutdown.

Approach	E_{active} (mJ)	E_{idle} (mJ)	E_{total} (mJ)	*Reduction* (%)
Nominal	1.96	1.02	2.98	—
DVddNoSH	1.74	0.93	2.68	10
DVddSH	1.75	0.62	2.56	14
DVddVbsNoSH	1.48	0.93	2.41	19
DVddVbsSH	1.50	0.70	2.30	23

TABLE III
RESULTS FOR THE GSM CODEC WITH SHUTDOWN

Approach	E_{active} (mJ)	E_{idle} (mJ)	E_{total} (mJ)	*Reduction* (%)
Nominal	26.93	6.94	33.87	—
DVddNoSH	20.78	4.83	25.61	25
DVddSH	20.83	0.20	22.53	34
DVddVbsNoSH	18.55	4.78	23.33	32
DVddVbsSH	19.85	0.20	21.56	37

TABLE IV
RESULTS FOR THE MMS SYSTEM WITH SHUTDOWN

In the previous experiments, communication energy has been ignored. Another set of experiments was performed on the two benchmarks in order to highlight the importance of combined processor and communication links' scaling. The GSM codec is considered to run on an architecture composed of 3 processors (with two voltage modes ($(1.8V, -0.1V)$ and $(1.0V, -0.6V)$)), communicating over a repeater-based shared bus. At the nomimal voltages, the communication accounts for 15% of the total energy consumption. Tab. V shows the resulting total energy consumptions for six different situations. The first column denotes the used voltage selection technique

Approach	VS type	E_{tot} (mJ)	Reduc. (%)
Nominal	—	2.273	—
CPU (V_{dd})	cont.	2.091	9
CPU (V_{dd},V_{bs})	cont.	1.831	20
Heu.CPU (V_{dd},V_{bs})	disc.	1.887	17
CPU+BUS (V_{dd},V_{bs})	cont.	1.665	27
Heu.CPU+BUS(V_{dd},V_{bs})	disc.	1.723	24

TABLE V
RESULTS FOR THE GSM CODEC CONSIDERING THE COMMUNICATION

and the second indicates if continuous or discrete voltages were considered. The third and fourth column give the energy consumption and achieved reduction in percentage for each scaling approach. For instance, according to the second row, the system dissipates an energy of $2.273\mu J$ at nominal voltage settings, i.e., without any voltage selection. This value serves as a baseline for the reductions indicated in the fourth column. The third and fourths row present the results of systems in which the bus remains unscaled while the processors are either V_{dd} or V_{dd} and V_{bs} scaled over a continuous range. As we can observe, savings of 9 and 20% are achieved. In order to adapt the continuous selected voltages towards the two discrete voltage settings at which the processor can possibly run, we apply our heuristic outlined in Section VII-D. The achieved reduction in the discrete case is 17% (row 5). Nevertheless, as shown by the values given in row 6, it is possible to further reduce the energy by scaling the repeater-based bus. Compared to the baseline, a saving of 27% is achieved. Using the discrete voltage heuristic, the final energy dissipation results in $1.723\mu J$, which is 24% below the unscaled system.

The MMS system is mapped on 4 processors that communicate over two repeater-based buses. At the nomimal voltages, the communication accounts for 25% of the total energy consumption. The results are presented in table VI.

Approach	VS type	E_{tot} (mJ)	Reduc. (%)
Nominal	—	35.01	—
CPU (V_{dd})	cont.	28.99	18
CPU (V_{dd},V_{bs})	cont.	26.05	26
Heu.CPU (V_{dd},V_{bs})	disc.	26.82	24
CPU+BUS (V_{dd},V_{bs})	cont.	22.94	35
Heu.CPU+BUS(V_{dd},V_{bs})	disc.	23.48	33

TABLE VI
RESULTS FOR THE MMS SYSTEM CONSIDERING THE COMMUNICATION

XI. CONCLUSIONS

Energy reduction techniques, such as supply voltage selection and adaptive body-biasing can be effectively exploited at the system-level. In this paper, we have investigated different alternatives of the combined supply voltage selection, adaptive body-biasing and processor shutdown problems at the system-level. These include the consideration of transition overheads as well as the discretization of the supply and threshold voltage levels. We have shown that nonlinear programming and mixed integer linear programming formulations can be used to solve these problems. Further, the NP-hardness of the discrete voltage selection case was shown, and a heuristic to efficiently solve the problem has been proposed. Similarly, if the shutdown of processors is considered, the problem becomes NP complete. Therefore, we have proposed an efficient heuristic to solve this problem. The voltage selection technique achieves additional efficiency by simultaneously scaling the voltages of processors and communication. We have investigated two alternatives, considering both buses with repeaters and fat wires. Several generated benchmark examples as well as two real-life applications were used to show the applicability of the introduced approaches.

In this paper we have focused on the voltage selection problem. The solutions presented and the heuristics proposed can be included in design space exploration frameworks that also perform other system level optimizations, such as task

mapping and scheduling. This has been demonstrated by integrating our work in the frameworks proposed in [42], [43].

ACKNOWLEDGEMENTS

Alexandru Andrei was supported by the Swedish Graduate School in Computer Science (CUGS). Petru Eles and Zebo Peng were supported by SSF through the STRINGENT excellence center. Marcus Schmitz and Bashir Al Hashimi were supported by EPSRC under grant GR/S95770.

REFERENCES

[1] T. Ishihara and H. Yasuura, "Voltage Scheduling Problem for Dynamically Variable Voltage Processors," in *Proc. Int. Symp. Low Power Electronics and Design (ISLPED'98)*, 1998, pp. 197–202.

[2] S. Martin, K. Flautner, T. Mudge, and D. Blaauw, "Combined Dynamic Voltage Scaling and Adaptive Body Biasing for Lower Power Microprocessors under Dynamic Workloads," in *Proc. ICCAD-02*, 2002, pp. 721–725.

[3] F. Yao, A. Demers, and S. Shenker, "A Scheduling Model for Reduced CPU Energy," *IEEE FOCS*, 1995.

[4] C. Kim and K. Roy, "Dynamic Vth Scaling Scheme for Active Leakage Power Reduction," in *Proc. Design, Automation and Test in Europe Conf. (DATE02)*, March 2002, pp. 163–167.

[5] S. Borkar, "Design Challenges of Technology Scaling," *IEEE Mirco*, pp. 23–29, July 1999.

[6] W. Kwon and T. Kim, "Optimal Voltage Allocation Techniques for Dynamically Variable Voltage Processors," *ACM Transactions on Embedded Computing Systems*, February 2005.

[7] F. Gruian and K. Kuchcinski, "LEneS: Task Scheduling for Low-Energy Systems Using Variable Supply Voltage Processors," in *Proc. ASP-DAC'01*, Jan 2001, pp. 449–455.

[8] J. Luo and N. Jha, "Power-profile Driven Variable Voltage Scaling for Heterogeneous Distributed Real-time Embedded Systems," in *Proc. VLSI'03*, 2003.

[9] M. Schmitz and B. M. Al-Hashimi, "Considering Power Variations of DVS Processing Elements for Energy Minimisation in Distributed Systems," in *Int. Symp. System Synthesis (ISSS'01)*, October 2001, pp. 250–255.

[10] Y. Zhang, X. Hu, and D. Chen, "Task Scheduling and Voltage Selection for Energy Minimization," in *Proc. IEEE DAC'02*, June 2002.

[11] D. Duarte, N. Vijaykrishnan, M. Irwin, H. Kim, and G. McFarland, "Impact of Scaling on The Effectiveness of Dynamic Power Reduction," in *Proc. ICCD*, Sept. 2002.

[12] I. Hong, G. Qu, M. Potkonjak, and M. B. Srivastava, "Synthesis Techniques for Low-Power Hard Real-Time Systems on Variable Voltage Processors," in *Proc. Real-Time Systems Symposium*, 1998.

[13] B. Mochocki, X. Hu, and G. Quan, "A Realistic Variable Voltage Scheduling Model for Real-Time Applications," in *Proc. ICCAD-02*, 2002, pp. 726–731.

[14] Y. Zhang, X. Hu, and D. Chen, "Energy Minimization of Real-time Tasks on Variable Voltage Processors with Transition Energy Overhead," in *Proc. ASP-DAC'03*, 2003, pp. 65–70.

[15] L. Yan, J. Luo, and N. Jha, "Joint dynamic voltage scaling and adpative body biasing for heterogeneous distributed real-time embedded systems,," *IEEE Trans. on Computer-Aided Design*, July 2005.

[16] R. G. R. Jejurikar, "Dynamic Slack Reclamation with Procrastination Scheduling in Real-Time Embedded Systems," in *Design Automation Conference*, Jun 2005.

[17] C. Svensson, "Optimum Voltage Swing on On-Chip and Off-Chip Interconnects," *IEEE J. Solid-State Circuits*, vol. 36, no. 7, pp. 1108–1112, July 2001.

[18] D. Sylvester and K. Keutzer, "Impact of Small Process Geometries on Microarchitectures in Systems on a Chip," *Proceedings of the IEEE*, vol. 89, no. 4, pp. 467–489, April 2001.

[19] Y. Ismail and E. Friedman, "Repeater Insertion in RLC Lines for Minimum Propagation Delay," in *Proc. ISCAS'99*, 1999, pp. 404–407.

[20] P. Kapur, G. Chandra, and K. Saraswat, "Power Estimation in Global Interconnects and its Reduction using a Novel Repeater Optimiazation Methodology," in *Proc. DAC'02*, 2002.

[21] L. Shang, L. Peh, and N. Jha, "Power-efficient Interconnection Networks: Dynamic Voltage Scaling with Links," *Comp. Arch. Letters*, vol. 1, no. 2, pp. 1–4, May 2002.

[22] G. Wei, J. Kim, D. Liu, S. Sidiropoulos, and M. Horowitz, "A Variable-Frequency Parallel I/O Interface with Adaptive Power-Supply Regulation," *IEEE J. Solid-State Circuits*, vol. 35, no. 11, pp. 1600–1610, Nov 2000.

[23] J. Liu, P. Chou, and N. Bagherzdeh, "Communication Speed Selection for Embedded Systems with Networked Voltage-Scalable Proceossors," in *Proc. CODES'02*, 2002.

[24] L. Benini, G. D. Micheli, E. Macii, D. Sciuto, and C. Silvano, "Address bus encoding techniques for system-level power optimization," in *Proc. DATE'98*, 1998, pp. 861–867.

[25] C.-T. Hsieh and M. Pedram, "Architectural Energy Optimization by Bus Splitting," *IEEE Trans. on CAD*, vol. 21, no. 4, pp. 408–414, April 2002.

[26] W. Fornaciari, D. Sciuto, and C. Silvano, "Power Estimation for Architectural Exploration of HW/SW Communication on System-Level Buses," in *Proc. 7th Int. Workshop Hardware/Software Co-Design (CODES'99)*, May 1999, pp. 152–156.

[27] G. Varatkar and R. Marculescu, "Communication-Aware Task Scheduling and Voltage Selection for Total System Energy Minimization," in *Proc. ICCAD'03*, 2003.

[28] A. Andrei, M. Schmitz, P. Eles, Z. Peng, and B. Al-Hashimi, "Overhead-Conscious Voltage Selection for Dynamic and Leakage Power Reduction of Time-Constraint Systems," in *Proc. Design, Automation and Test in Europe Conf. (DATE04)*, Feb 2004, pp. 518–523.

[29] A. Andrei, M. Schmitz, P. Eles, Z. Peng, and B. A. Hashimi, "Simultaneous Communication and Processor Voltage Scaling for Dynamic and Leakage Energy Reduction in Time-Constrained Systems," in *ICCAD*, Nov 2004.

[30] ——, "Quasi-Static Voltage Scaling for Energy Minimization with Time Constraints," in *DATE*, Nov 2005.

[31] A. P. Chandrakasan and R. W. Brodersen, *Low Power Digital CMOS Design*. Kluwer Academic Publisher, 1995.

[32] "Intel® XScale™ Core, Developer's Manual," December 2000.

[33] "Mobile AMD Athlon™4, Processor Model 6 CPGA Data Sheet," November 2000, publication No 24319 Rev E.

[34] A. Andrei, M. Schmitz, P. Eles, and Z. Peng, "Overhead-Conscious Voltage Selection for Dynamic and Leakage Energy Reduction of Time-Constrained Systems," Linkoping University, Department of Computer and Information Science, Sweden," Technical Report, Mar. 2006.

[35] Y. Nesterov and A. Nemirovskii, *Interior-Point Polynomial Algorithms in Convex Programming*. Studies in Applied Mathematics, 1994.

[36] P. De, E. Dunne, J. Ghosh, and C. Wells, "Complexity of the Discrete Time-Cost Tradeoff problem for Project Networks," *Operations Research*, vol. 45, no. 2, pp. 302–306, March 1997.

[37] P. Caputa and C. Svensson, "Low-Power, Low-Latency Global Interconnects," in *Proc. IEEE ASIC/SOC'02*, 2002, pp. 394–398.

[38] "Mosek optimization software," http://www.mosek.com.

[39] K. Banerjee and A. Mehrotra, "A Power-Optimal Repeater Insertion Methodology for Global Interconnects in Nanometer Designs," *IEEE Trans. on Electron Devices*, vol. 49, no. 11, pp. 2001–2006, No 2002.

[40] M. Schmitz, B. Al-Hashimi, and P. Eles, *System-Level Design Techniques for Energy-Efficient Embedded Systems*. Kluwer Academic Publisher, 2004.

[41] J. Hu and R. Marculescu, "Energy-aware mapping for tile-based NoC architectures under performance constraints," in *Proc. ASPDAC'03*, 2003, pp. 233 – 239.

[42] M. Ruggiero, P. Gioia, G. Alessio, L. B. M. Milano, D. Bertozzi, and A. Andrei, "A Cooperative, Accurate Solving Framework for Optimal Allocation, Scheduling and Frequency Selection on Energy-Efficient MPSoCs," in *SOC*, Nov 2006.

[43] M. Schmitz, B. A. Hashimi, and P. Eles, "Cosynthesis of Energy-Efficient Multimode Embedded Systems With Consideration of Mode-Execution Probabilities," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 2, pp. 153–169, Feb 2005.