Energy Optimization of Multiprocessor Systems on Chip by Voltage Selection

Alexandru Andrei, *Student Member, IEEE*, Petru Eles, *Member, IEEE*, Zebo Peng, *Senior Member, IEEE*, Marcus T. Schmitz, and Bashir M. Al Hashimi, *Senior Member, IEEE*

***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 multiprocessor 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.**

***Index Terms*—Energy management, power minimization, real-time systems, voltage selection.**

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 nonuniform over time. This introduces slack times during which the system can reduce its performance to save energy. Two system-level approaches that allow an energy/performance tradeoff during runtime of the application are dynamic voltage selection (DVS) [1]–[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 $V_{\rm dd}$, ABB is effective in reducing the leakage power by scaling down frequency and increasing the threshold voltage $V_{\rm th}$ 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-submicrometer 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 [4], [5]. 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 online and offline techniques. In the following, we restrict ourselves to the offline techniques since the presented approaches fall into this category, where the scaled supply voltages are calculated at design time and then applied at runtime according to the precalculated 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 nonuniform 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]–[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 previously mentioned approaches scale only the supply voltage $V_{\rm dd}$ and neglect leakage power consumption, Kim and Roy [4] proposed an adaptive body-biasing approach (in their work referred to as dynamic $V_{\rm th}$ 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 multiprocessor systems under realtime constraints. Furthermore, as opposed to our work, the techniques mentioned *neglect* the energy and time overheads imposed by voltage transitions.

Manuscript received May 4, 2006; revised September 19, 2006. A. Andrei was supported by the Swedish Graduate School in Computer Science (CUGS). P. Eles and Z. Peng were supported by Swedish Foundation for Strategic Research (SSF) through the STRINGENT Excellence Center. M. T. Schmitz and B. M. Al Hashimi were supported by Engineering and Physical Sciences Research Council (EPSRC) under Grant GR/S95770.

A. Andrei, P. Eles, and Z. Peng are with the Department of Computer and Information Science, Linköping SE-58 183, Sweden (e-mail: alean@ida.liu.se).

M. T. Schmitz is with Diesel Systems for Commercial Vehicles, Robert Bosch GmbH, Stuttgart 70469, Germany.

B. M. Al-Hashimi is with the Computer Engineering Department, Southampton University, Southampton, SO 17 1BJ, U.K.

Digital Object Identifier 10.1109/TVLSI.2007.891101

Noticeable exceptions are [12]–[14], yet their algorithms ignore leakage power dissipation and body biasing, and further they do not guarantee optimality. In this paper, 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 demonstrate 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: 1) the insertion of *repeaters*; 2) the usage of *fat wires*. In principle, repeaters split long wires into shorter (faster) segments [18]–[20] and fat wires reduce the wire resistance [17], [18]. Techniques for the determination of the optimal quantity of repeaters are introduced in [19] and [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] and [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 simultaneous processor and communication voltage selection in [28], [29], and [30].

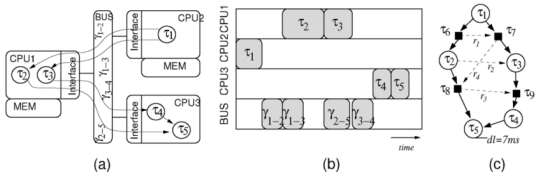


Fig. 1. System models. (a) Target architecture with mapped task graph. (b) Multiple component schedule. (c) Extended TG.

As mentioned earlier, in this paper, we will concentrate on offline voltage selection techniques that make use of the static slack existing in the application. In [31], 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 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, field-programmable gate arrays (FPGAs), and application specified integrated circuits (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 runtime.

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 a 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: 1) dynamic power $P_{\rm dyn}$, which is dissipated whenever active computations are carried out (switching of logic states) and 2) leakage power $P_{\rm leak}$ which is consumed whenever the circuit is powered, even if no computations are performed. The dynamic power is expressed by [32], [2]

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

where $C_{\rm eff}$, f, and $V_{\rm dd}$ 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, (< 65 nm) it is expected that leakage will account for a significant part of the total power. The leakage power is given by [2]

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

where $V_{\rm bs}$ is the body-bias voltage and $I_{\rm Ju}$ 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 $V_{\rm bs}$ than by $V_{\rm dd}$, 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 [32], [2]

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

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 $V_{\rm th1}$ 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 $\epsilon_{k,j}$ and delay $\delta_{k,j}$ overheads, when switching from $V_{{\rm dd}_k}$ to $V_{{\rm dd}_j}$ and from $V_{{\rm bs}_k}$ to $V_{{\rm bs}_j}$, are given by [2]

$$\epsilon_{k,j} = C_r \cdot |V_{{\rm dd}_k} - V_{{\rm dd}_j}|^2 + C_s \cdot |V_{{\rm bs}_k} - V_{{\rm bs}_j}|^2 \tag{4}$$

$$\delta_{k,j} = \max(p_{\rm Vdd} \cdot |V_{{\rm dd}_k} - V_{{\rm dd}_j}|, p_{\rm Vbs} \cdot |V_{{\rm bs}_k} - V_{{\rm bs}_j}|) \tag{5}$$

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

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: $m_z = (V_{{\rm dd}_z}, V_{{\rm bs}_z})$. As a result, an execution mode has an associated frequency and power consumption (dynamic and leakage) that can be calculated using (3) and, respectively, (1) and (2). Upon a mode change, the corresponding delay and energy penalties are computed using (4) and (5).

Tasks that are mapped on different processors communicate over one or more shared buses. In Sections IV–VIII, 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 depending proportionally on the amount of communication. For simplicity of the explanations, in Sections IV–VIII, 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 $V_{\rm dd}$ scaling only (i.e., $V_{\rm bs}$ is kept constant), the second schedule corresponds to the simultaneous scaling of $V_{\rm dd}$ and $V_{\rm bs}$. 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 (< 70 nm) which leads to a leakage power consumption close to 40% of the total power consumed (at the mode with the highest performance).

Let us consider the first schedule in which the tasks are executed either at $V_{\rm dd1} = 1.8$ V, or $V_{\rm dd2} = 1.5$ V, while $V_{\rm bs1}$ and $V_{\rm bs2}$ are kept at 0 V. In accordance, the system dissipates $P_{\rm dyn1} = 100$ mW and $P_{\rm leak1} = 75$ mW in mode 1 running at 700 MHz, while $P_{\rm dyn2} = 49$ mW and $P_{\rm leak2} = 45$ mW in mode 2 running at 525 MHz, as observable from the figure. We

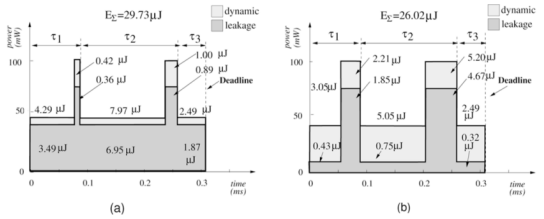


Fig. 2. Influence of $V_{\rm bs}$ scaling. (a) $V_{\rm dd}$ scaling only. (b) Simultaneous $V_{\rm dd}$ and $V_{\rm bs}$ scaling.

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 and 16.17 μJ, respectively. This results in a total energy consumption of 29.73 μJ.

Consider now the schedule given in Fig. 2(b), where tasks are executed at two different voltage settings for $V_{\rm dd}$ and $V_{\rm bs}$ [$m_1 =$ (1.8 V, 0 V) and $m_2 =$ (1.5 V, -0.4 V)]. Since the voltage settings for mode m_1 did not change, the system runs at 700 MHz and dissipates $P_{\rm dyn1} = 100$ mW and $P_{\rm leak1} = 75$ mW. In mode m_2 the system performs at 480 MHz and dissipates $P_{\rm dyn2} = 49$ mW and $P_{\rm leak2} = 5$ mW. There are two main differences to observe compared to the schedule in Fig. 2(a). First, 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 -0.4 V [see (2)]. Second, the high voltage mode m_1 is active for a longer time; this can be explained by the fact that scaling $V_{\rm bs}$ during mode m_2 requires the reduction of the operational frequency [see (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 to 18.0 μJ, compared to the first schedule, the leakage was reduced from 13.56 to 8.02 μJ. The overall energy dissipation is 26.02 μJ, a reduction by 12.5%. This example illustrates the advantage of simultaneous $V_{\rm dd}$ and $V_{\rm bs}$ scaling compared to $V_{\rm dd}$ scaling only.

B. Considering the Transition Overheads

We consider a single processor system that offers three voltage modes, $m_1 =$ (1.8 V, -0.3 V), $m_2 =$ (1.5 V, -0.45 V), and $m_3 =$ (1.2 V, -0.8 V), where $m_z = (V_{{\rm dd}_z}, V_{{\rm bs}_z})$. The rail and substrate capacitance are given as $C_r = 10$ μF and $C_s = 40$ μF. The processor needs to execute two consecutive tasks (τ_1 and τ_2) with a deadline of 0.225 ms. 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 $E = 9 + 15 + 4.5 + 7.5 = 36$ μJ. 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 [33], [34], 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 $10\,\mu{\rm F}\cdot(1.8\,{\rm V} - 1.5\,{\rm V})^2 + 40\,\mu{\rm F}\cdot(0.3\,{\rm V} - 0.45\,{\rm V})^2 = 1.8\,\mu{\rm J}$,

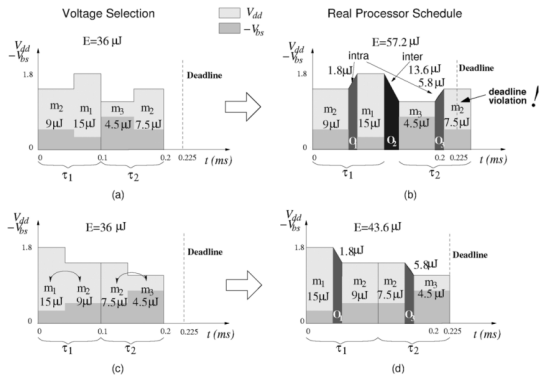


Fig. 3. Influence of transition overheads. (a) Before reordering, without overheads. (b) Before reordering, with overheads. (c) After reordering, without overheads. (d) After reordering, with overheads.

based on (4). Similarly, the energy overheads for transitions O_2 and O_3 can be calculated as 13.6 μJ and 5.8 μJ, respectively. The overall energy dissipation of the schedule from Fig. 3(b) accumulates to $36 + 1.8 + 13.6 + 5.8 = 57.2$ μJ.

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 $E = 43.6$ μJ, 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 ${\rm dl}_i$, its worst case number of clock cycles to be executed NC_i and the switched capacitance $C_{{\rm eff}_i}$ are given. Each processor can vary its supply voltage $V_{\rm dd}$ *and* body-bias voltage $V_{\rm bs}$ within certain continuous ranges (for the continuous problem), or, within a set of discrete voltage pairs $m_z = \{(V_{{\rm dd}_z}, V_{{\rm bs}_z})\}$ (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 $V_{\rm dd}$ and/or $V_{\rm bs}$, a transition overhead in terms of energy and time is required [see (4) and (5)].

For reasons of clarity, we introduce the following four distinctive problems which will be considered in this paper: 1) continuous voltage selection with no consideration of transition overheads (CNOH); 2) continuous voltage selection with consideration of transition overheads (COH); 3) discrete voltage selec-

tion with no consideration of transition overheads (DNOH); and 4) 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:

$$\text{Minimize} \quad \sum_{k=1}^{|\mathcal{T}|} \Big(\underbrace{NC_k \cdot C_{\text{eff}_k} \cdot V_{\text{dd}_k}^2}_{E_{\text{dyn}_\text{k}}} + \underbrace{L_g (K_3 \cdot V_{\text{dd}_k} \cdot e^{K_4 \cdot V_{\text{dd}_k}} \cdot e^{K_5 \cdot V_{\text{bs}_k}} + I_{\text{Ju}} \cdot |V_{\text{bs}_k}|) \cdot t_k}_{E_{\text{leak}_\text{k}}} \Big) \tag{6}$$

subject to

$$t_k = NC_k \cdot \frac{(K_6 \cdot L_d \cdot V_{\text{dd}_k})}{((1+K_1) \cdot V_{\text{dd}_k} + K_2 \cdot V_{\text{bs}_k} - V_{\text{th}_1})^\alpha} \tag{7}$$

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

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

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

$$V_{\text{dd}_\text{min}} \leq V_{\text{dd}_k} \leq V_{\text{dd}_\text{max}} \quad \text{and} \quad V_{\text{bs}_\text{min}} \leq V_{\text{bs}_k} \leq V_{\text{bs}_\text{max}}. \tag{11}$$

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 (6). The task execution time has to be equivalent to the number of clock cycles of the task multiplied by the circuit delay for a particular V_{dd_k} and V_{bs_k} setting, as expressed by (7). Given the execution time of the tasks, it becomes possible to express the precedence constraints between tasks [see (8)], i.e., a task τ_l can only start its execution after all its predecessor tasks τ_k have finished their execution $(D_k + t_k)$. Predecessors of task τ_l are all tasks τ_k for which there exists an edge $(k,l) \in \mathcal{E}$ in the mapped and scheduled task graph. Similarly, tasks with deadlines have to be completed $(D_k + t_k)$ before their deadlines dl_k [see (9)]. Task start times have to be positive [see (10)] and the imposed voltage ranges should be respected [see (11)]. It should be noted that the objective [see (6)] as well as the task execution time [see (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 $\epsilon > 0$), [35].

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:

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

subject to

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

$$\delta_{k,j} = \max(p_{\text{Vdd}} \cdot |V_{\text{dd}_k} - V_{\text{dd}_j}|, p_{\text{Vbs}} \cdot |V_{\text{bs}_k} - V_{\text{bs}_j}|). \tag{14}$$

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

VII. Optimal Discrete Voltage Selection

The approaches presented in Section VI 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.

The details of the proof are given in [30]. 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][1] impossible, which states that the discrete voltage selection problem (considered in [6] without body biasing and overheads) can be solved optimally in polynomial time.

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

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

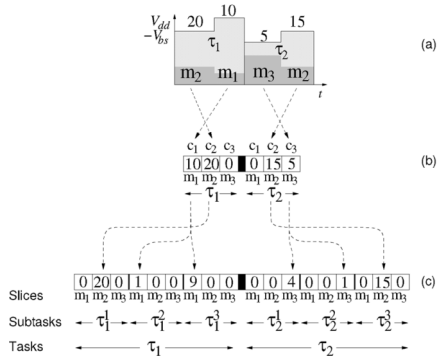


Fig. 4. Discrete mode model. (a) Schedule and mode execution order. (b) Tasks and clock cycles in each mode (mode execution order is not captured). (c) Solution vector with division (mode execution order is captured).

is characterized by a voltage pair $(V_{\mathrm{dd}_m}, V_{\mathrm{bs}_m})$, which determines the operational frequency f_m, the normalized dynamic power P_{dnom_m}, and the leakage power dissipation $P_{\mathrm{leak_m}}$. The frequency and the leakage power are given by (3) and (2), respectively. The normalized dynamic power is given by $P_{\mathrm{dnom}_m} = f_m \cdot V_{\mathrm{dd}_m}^2$. Accordingly, the dynamic power of a task τ_k, operating in mode m, is computed as $C_{\mathrm{eff}_k} \cdot P_{\mathrm{dnom}_m}$. Based on these definitions, the problem is formulated as follows:

Minimize

$$\sum_{k=1}^{|\mathcal{T}|} \sum_{m \in \mathcal{M}} (C_{\mathrm{eff}_k} \cdot P_{\mathrm{dnom}_m} \cdot t_{k,m} + P_{\mathrm{leak}_m} \cdot t_{k,m}) \tag{15}$$

subject to

$$D_k + \sum_{m \in \mathcal{M}} t_{k,m} \leq \mathrm{dl}_k \tag{16}$$

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

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

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

The total energy consumption, expressed by (15), is given by two sums. The inner sum indicates the energy dissipated by an individual task τ_k, depending on the time $t_{k,m}$ spent in each mode m. 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 Fig. 4(a) and (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 $(\sum_{m \in \mathcal{M}} t_{k,m})$. Equation (16) ensures that all the deadlines are met and (17) maintains the correct execution order given by the precedence relations. The relation between execution time and number of clock cycles as well as the requirement to execute all clock cycles of a task are expressed in (18). Additionally, task start times D_k and task execution times have to be positive [see (19)].

C. Discrete Voltage Selection With Overheads (DOH)

The details regarding the incorporation of transition overheads into the MILP formulation from Section VII-B are presented in [28]. The order in which the modes are activated has an influence on the transition overheads, as we have illustrated in Section IV-B. We introduce the following extensions needed in order to take both delay and energy overheads into account. Given m operational modes, the execution of a single task τ_k can be subdivided into m subtasks $\tau_k^s, s = 1, \ldots, m$. Each subtask is executed in one and only one of the m modes. Subtasks are further subdivided into m slices, each corresponding to a mode. This results in $m \cdot m$ 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 nine cycles are executed by the last subtask in mode m_1; τ_2 executes in its first subtask four clock cycles in mode m_3, one 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.

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. 5(a). If the application runs on a single processor system at the lowest speed, it still finishes before the deadline, as depicted in Fig. 5(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 shutting 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. 5(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.

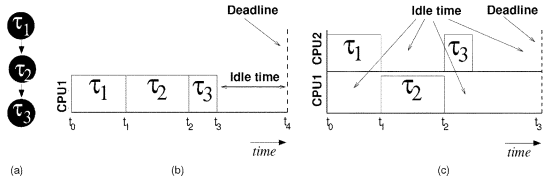


Fig. 5. Schedules with idle times. (a) Task graph. (b) Single processor. (c) Multiprocessor.

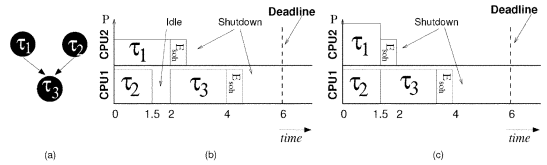


Fig. 6. Voltage selection with shutdown. (a) Task graph. (b) Voltage scaling and shutdown. (c) Voltage scaling + shutdown.

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 Section IX.

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 orthogonal. Let us consider an application consisting of three tasks, τ_1,, τ_2, and τ_3, as in Fig. 6(a). The tasks are mapped on two processors CPU_1 and CPU_2. The resulting schedule, after performing voltage selection is depicted in Fig. 6(b), with all three tasks running at the lowest speeds. Task τ_1 is running for 2 ms with 200 mW, while τ_2 and τ_3 run at 400 mW for 1.5 and 2 ms, respectively. 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 μJ on CPU_2. We notice the idle interval of 0.5 ms on CPU_2, between the executions of τ_2 and τ_3. The idle power on CPU_2 is 250 mW, resulting in an energy consumption of 125 μJ. 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 1.5 ms. The power consumption that corresponds to this frequency is 300 mW. 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 μJ.

A. Processor Shutdown: Problem Complexity

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

The proof is given in [30]. 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.[2] The following gives the modified nonlinear programming formulation (CVSSH):

$$\text{Minimize} \quad \sum_{k=1}^{|\mathcal{T}|} E_{\mathrm{dyn}_k} + \sum_{k=1}^{|\mathcal{T}|} E_{\mathrm{leak}_k} + \underbrace{\sum_{k=1}^{|\mathcal{T}|} xi_k \cdot t_{\mathrm{idle}_k} \cdot P_{\mathrm{idle}_k} + xs_k \cdot (E_{\mathrm{soh}_k} + t_{\mathrm{off}_k} \cdot P_{\mathrm{off}_k})}_{E_{\mathrm{idle}} + E_{\mathrm{off}}} \tag{20}$$

subject to

$$t_k = NC_k \cdot \frac{(K_6 \cdot L_d \cdot V_{\mathrm{dd}_k})}{((1+K_1)\cdot V_{\mathrm{dd}_k} + K_2 \cdot V_{\mathrm{bs}_k} - V_{\mathrm{th}_1})^{\alpha}} \tag{21}$$

$$D_k + t_k \le D_l \quad \forall (k,l) \in \mathcal{E} - \mathcal{E}^{\bullet} \tag{22}$$

$$D_k + t_k + xi_k \cdot t_{idle_k} = D_l \quad \forall (k,l) \in \mathcal{E}^{\bullet} \tag{23}$$

$$D_k + t_k + xs_k \cdot T_{\mathrm{soh}_k} + t_{\mathrm{off}_k} = D_l \quad \forall (k,l) \in \mathcal{E}^{\bullet} \tag{24}$$

$$xi_k + xs_k = 1 \quad \forall \tau_k \tag{25}$$

$$D_k + t_k \le \mathrm{dl}_k \quad \forall \tau_k \text{with dl} \tag{26}$$

$$xi_k, xs_k \in \{0,1\} \tag{27}$$

$$D_k \ge 0 \quad \text{and} \quad V_{\mathrm{dd}_{\min}} \le V_{\mathrm{dd}_k} \le V_{\mathrm{dd}_{\max}} \quad \text{and} \, V_{\mathrm{bs}_{\min}} \le V_{\mathrm{bs}_k} \le V_{\mathrm{bs}_{\max}}. \tag{28}$$

There are two noticeable differences between this formulation and the one in Section VI-A: the inclusion in the objective (20) of the energy spent during idle and shutdown intervals and (24) and (23) 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 (20), (23), and (24) 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 Section VIII, 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 volt-

[2]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

```
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, toff_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_off_k, tidle_k)
```

Fig. 7. Voltage selection with shutdown heuristic.

ages 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. 7. 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 ($xs_k = 0$ 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 (lines 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 (lines 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 the following 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 $t_{\mathrm{idle}_k}^1$, that is too small to provide savings via shutdown: $t_{\mathrm{idle}_k}^1 \cdot P_{\mathrm{idle}_k} < E_{\mathrm{soh}_k}$. The total energy consumed in this case is $E_k^1 + t_{\mathrm{idle}_k}^1 \cdot P_{\mathrm{idle}_k}$. 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 $E_k^2 + E_{\mathrm{soh}_k}$. Since $E_k^2 > E_k^1$ and $E_{\mathrm{soh}_k} > t_{\mathrm{idle}_k}^1 \cdot P_{\mathrm{idle}_k}$, 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. 6. 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 $t_{\mathrm{idle}_{k,l,m}}$ 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 second, 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 [29] 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

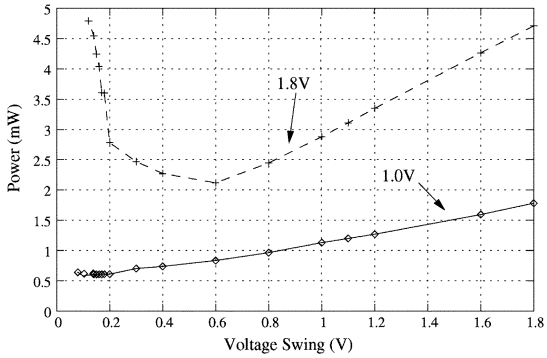


Fig. 8. Optimum swing on a fat wire bus.

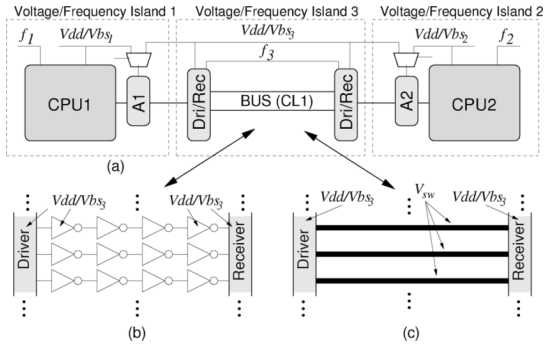


Fig. 9. Interconnect structures. (a) Interconnect structure. (b) Repeater-based bus. (c) Fat wire-based bus.

solvable in polynomial time, while for the discrete voltage selection case, we use a heuristic similar to the one presented in [29]. 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

Repeaters are simple CMOS invertors introduced on long wires in order to speed-up the communication time. The same voltage selection techniques as in the case of processors can be applied for buses implemented with repeaters [29].

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. 8 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 70-nm 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 1.0 V, the lowest bus power dissipation (0.55 mW) is achieved by a voltage swing of 0.14 V. Let us assume that the voltages of the driver and receiver are changed during runtime to 1.8 V 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 0.14 V, the power dissipation on the bus will be 4.5 mW. However, inspecting the plot reveals that it is possible to reduce the bus power dissipation by changing the voltage swing from 0.14 to 0.6 V. At this voltage swing, the bus dissipates a power of 2.2 mW, 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 1.8 to 1.0 V. Keeping the swing at 0.6 V results in a power of 0.83 mW, which is, compared to the optimal 0.55 mW at 0.14 V, 33% higher than necessary.

C. Communication Models

We consider a bus-based communication system as in Fig. 9. 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

$$E_{\mathrm{adapter}} = C_{\mathrm{adapter}} \cdot (V_{\mathrm{dd}_{\mathrm{CPU}}} - V_{\mathrm{dd}_{\mathrm{bus}}})^2. \tag{29}$$

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

$$E_{\mathrm{comm}} = E_{\mathrm{adapter}_1} + E_{\mathrm{bus}} + E_{\mathrm{adapter}_2}. \tag{30}$$

Feature size scaling in deep-submicrometer 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: 1) the usage of repeaters [19], [20] and 2) the usage of fat wires [17], [18]. The bus energy E_{bus} in (30) 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. Sylvester and Keutzer [18] estimate an increasing number of repeaters with technology scaling down. For instance, up to 138 repeaters are used in 50-nm technology for a corner-to-corner wire with a die size of 750 mm^2. Repeaters are implemented as simple CMOS inverter circuits [Fig. 9(b)]. In accordance, the power dissipated by a bus implemented with repeaters is given by

$$P_{\mathrm{rep}} = N \cdot (\underbrace{s_\tau \cdot C_{\mathrm{rep}} \cdot V_{\mathrm{dd}}^2 \cdot f}_{P_{\mathrm{dyn}}} + \underbrace{V_{\mathrm{dd}} \cdot K_3 \cdot e^{K_4 \cdot V_{\mathrm{dd}}} \cdot e^{K_5 \cdot V_{\mathrm{bs}}} + |V_{\mathrm{bs}}| \cdot I_{\mathrm{Ju}}}_{P_{\mathrm{leak}}}) \tag{31}$$

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 (3) to approximate the operational frequency f of the bus. The execution time of a communication $\tau \in \mathcal{K}$ is given by

$$t = \left\lceil \frac{NB_\tau}{W_{\text{bus}}} \right\rceil \cdot \frac{1}{f} \tag{32}$$

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). According to (31) and (32), the bus energy dissipation is given by $E_{\text{bus}} = P_{\text{rep}} \cdot t$. 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 (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 $L \cdot R_w/Z_0 < 2\ln 2$ (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 4-mm-long wire in 180-nm technology, Caputa and Svensson [36] obtained a fat wire width of 2μm 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

$$P_{\text{dri}_{\text{dyn}}} = s_\tau \cdot f \cdot (C_{\text{dri}} + C_w) \cdot V_{\text{dd}}^2 \tag{33}$$

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

$$P_{\text{dri}_{\text{dyn}}} = \begin{cases} s_\tau \cdot f \cdot (C_{\text{dri}} + C_w) \cdot V_{\text{dd}} \cdot V_{\text{sw}}, & \text{if } V_{\text{sw}} \text{ is generated on chip} \\ s_\tau \cdot f \cdot (C_{\text{dri}} + C_w) \cdot V_{\text{sw}}^2, & \text{otherwise.} \end{cases} \tag{34}$$

The driver dissipates a nonnegligible leakage power

$$P_{\text{dri}_{\text{leak}}} = L_g \cdot (V_{\text{dd}} \cdot K_3 \cdot e^{K_4 V_{\text{dd}}} \cdot e^{K_5 V_{\text{bs}}} + |V_{\text{bs}}| \cdot I_{\text{Ju}}). \tag{35}$$

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

$$P_{\text{rec}_{\text{dyn}}} = s_\tau \cdot f \cdot C_{\text{rec}} \cdot V_{\text{dd}}^2 \tag{36}$$

$$P_{\text{rec}_{\text{leak}}} = L_g \cdot (V_{\text{dd}} \cdot K_3 \cdot e^{K_4 \cdot V_{\text{dd}}} \cdot e^{K_L \cdot (V_{\text{dd}}/2 - V_{\text{sw}}/2)} \cdot e^{K_5 \cdot V_{\text{bs}}} + |V_{\text{bs}}| \cdot I_{\text{Ju}}). \tag{37}$$

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, (34)]. In order to find the most efficient solution we need to find an appropriate voltage swing that minimizes the total bus power $P_{\text{bus}} = P_{\text{dri}_{\text{dyn}}} + P_{\text{dri}_{\text{leak}}} + P_{\text{rec}_{\text{dyn}}} + P_{\text{rec}_{\text{leak}}}$. Using the optimal voltage swing can significantly reduce the power consumption of the bus [36], [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 (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. 8). Since these settings are dynamically changed during runtime via voltage selection, the value of the optimal voltage swing changes as well during runtime, 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 (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

$$\begin{aligned} \epsilon_{k,j} &= C_{\text{wr}} \cdot (V_{\text{sw}_k} - V_{\text{sw}_j})^2 \\ \delta_{k,j} &= p_{V\text{sw}} \cdot |V_{\text{sw}_k} - V_{\text{sw}_j}| \end{aligned} \tag{38}$$

where C_{wr} is the wire power rail capacitance and $p_{V\text{sw}}$ 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 $m_z = \{(V_{\text{dd}_z}, V_{\text{bs}_z})\}$ (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.

1) 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 $m_z = \{(V_{\text{dd}_z}, V_{\text{bs}_z})\}$, 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.

2) Fat Wire Implementation: If communication is performed over fat wires, it is necessary to dynamically adapt the voltage swing at which data is transferred. 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 $n_z = \{V_{\mathrm{sw}_z}\}$ and for the bus drivers and receiver the set is $m_z = \{(V_{\mathrm{dd}_z}, V_{\mathrm{bs}_z})\}$. 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

$$\underbrace{\sum_{k}^{|\Pi|} E_{\mathrm{dyn}_k} + E_{\mathrm{leak}_k}}_{\text{computation}} + \underbrace{\sum_{k}^{|\mathcal{K}|} E_{\mathrm{dyn_k}} + E_{\mathrm{leak_k}}}_{\text{communication}} + \underbrace{\sum_{(k,j)\in\mathcal{E}^\bullet} \epsilon_{k,j}}_{\text{overhead}} \tag{39}$$

subject to

$$t_k = \begin{cases} NC_k \cdot \frac{(K_6 \cdot L_d \cdot V_{\mathrm{dd}_k})}{((1+K_1)\cdot V_{\mathrm{dd}_k} + K_2 \cdot V_{\mathrm{bs}_k} - V_{\mathrm{th}_1})^\alpha}, & \text{if } \tau_k \in \Pi \\ \left\lceil \frac{NB_k}{W_{\mathrm{bus}}} \right\rceil \cdot \frac{(K_6 \cdot L_d \cdot V_{\mathrm{dd}_k})}{((1+K_1)\cdot V_{\mathrm{dd}_k} + K_2 \cdot V_{\mathrm{bs}_k} - V_{\mathrm{th}_1})^\alpha}, & \text{if } \tau_k \in \mathcal{K} \end{cases} \tag{40}$$

$$D_k + t_k \le D_l \quad \forall (k,l) \in \mathcal{E} \tag{41}$$

$$D_k + t_k + \delta_{k,l} \le D_l \quad \forall (k,l) \in \mathcal{E}^\bullet \tag{42}$$

$$D_k + t_k \le \mathrm{dl}_k \quad \forall \tau_k \in \Pi \text{ with a deadline} \tag{43}$$

$$D_k \ge 0 \tag{44}$$

$$V_{\mathrm{dd_{min}}} \le V_{\mathrm{dd}_k} \le V_{\mathrm{dd_{max}}} \tag{45}$$

$$V_{\mathrm{bs_{min}}} \le V_{\mathrm{bs}_k} \le V_{\mathrm{bs_{max}}} \tag{46}$$

$$V_{\mathrm{sw_{min}}} \le V_{\mathrm{sw}_k} \le V_{\mathrm{sw_{max}}}. \tag{47}$$

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 [see (39)], 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 (derived from the equations discussed in Section III)

$$E_{\mathrm{dyn_k}} = \begin{cases} NC_k \cdot s_k \cdot C_{\mathrm{eff}_k} \cdot V_{\mathrm{dd}_k}^2, & \text{if } \tau_k \in \Pi \\ \sum^N \left\lceil \frac{NB_k}{W_{\mathrm{bus}}} \right\rceil \cdot s_k \cdot C_{\mathrm{rep}} \cdot V_{\mathrm{dd}_k}^2, & \text{if } \tau_k \in \mathcal{K} \text{ on repeaters} \\ \left\lceil \frac{NB_k}{W_{\mathrm{bus}}} \right\rceil \cdot s_k \cdot C_{\mathrm{fat}} \cdot V_{\mathrm{dd}_k} \cdot V_{\mathrm{sw}_k}, & \text{if } \tau_k \in \mathcal{K} \text{ on fat wires (intern)} \\ \left\lceil \frac{NB_k}{W_{\mathrm{bus}}} \right\rceil \cdot s_k \cdot C_{\mathrm{fat}} \cdot V_{\mathrm{sw}_k}^2, & \text{if } \tau_k \in \mathcal{K} \text{ on fat wires (extern)} \end{cases} \tag{48}$$

where $C_{\mathrm{rep}} = C_d + C_w + C_g$ and $C_{\mathrm{fat}} = C_{\mathrm{dri}} + C_w + C_{\mathrm{rec}}$ 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

$$E_{\mathrm{leak_k}} = L_g(K_3 \cdot V_{\mathrm{dd}_k} \cdot e^{K_4 \cdot V_{\mathrm{dd}_k}} \cdot e^{K_5 \cdot V_{\mathrm{bs}_k}} + I_{\mathrm{Ju}} \cdot |V_{\mathrm{bs}_k}|) \cdot t_k. \tag{49}$$

For fat wire-based buses, we need to additionally account for the leakage in the receiver [see (35) and (37)], given by

$$E_{\mathrm{leak}_k} = (P_{\mathrm{dri_{leak}}} + P_{\mathrm{rec_{leak}}}) \cdot t_k. \tag{50}$$

The energy overhead due to voltage transitions is given by (4) and (38).

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 [29] in order to translate the computed continuous voltages into the discrete ones available for the processors and buses.

X. Experimental Results

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 [37] and [38]. Experimental results using randomly generated task graphs have been presented in [28]–[30].

The GSM voice codec consists of 87 tasks and is considered to run on an architecture composed of three processing elements with two voltage modes [(1.8 V, −0.1 V) and (1.0 V, −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, $p_{\mathrm{Vdd}} = 10\ \mu$s/V, and $p_{\mathrm{Vbs}} = 10\ \mu$s/V. Table 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

TABLE I
OPTIMIZATION RESULTS FOR THE GSM CODEC

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 II
OPTIMIZATION RESULTS FOR THE MMS SYSTEM

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

consumption can be reduced by 22%, yet taking the overheads into account 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 [(1.8 V, 0.0 V), (1.6 V, −0.8), (1.3 V, −0.9), and (1.0 V, −0.9)]. At the highest voltage mode, the application reveals a deadline slack close to 40%. Table II shows the results in terms of dynamic $E_{\rm dyn}$, leakage $E_{\rm leak}$, overhead ϵ, and total $E_{\rm active}$ energy (Columns 2–5). As with the GSM, the consideration of overheads achieves higher energy savings (22.9% for the $V_{\rm dd}$-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 $(E_{\rm active})$, idle and total energy $(E_{\rm idle})$ consumption. The overheads for a shutdown operation are estimated in [16] as $E_{\rm soh} = 300\ \mu$J and $t_{\rm soh} =$1 ms.

TABLE III
RESULTS FOR THE GSM CODEC WITH 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 IV
RESULTS FOR THE MMS SYSTEM 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

If we use these values for the GSM voice codec, we can not perform any shutdown, due to the little amount of slack available after voltage selection. If we consider lower shutdown overheads ($E_{\rm soh} = 90\ \mu$J and $t_{\rm soh} = 0.3$ ms), 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 1.48 to 1.50 mJ), while the idle energy is reduced (from 0.93 to 0.70 mJ). This means that a situation similar to the one described in Fig. 6 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 $(E_{\rm total})$ and the sum of active $(E_{\rm active})$ and idle $(E_{\rm idle})$ 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] ($E_{\rm soh} = 300\ \mu$J and $t_{\rm soh} = 1$ ms). 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 $V_{\rm dd}$ and $V_{\rm bs}$ 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 five 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 [31] an online voltage selection technique that can make use of dynamic slack. The technique is based on an offline calculation of

TABLE V
RESULTS FOR THE GSM CODEC CONSIDERING THE COMMUNICATION

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 VI
RESULTS FOR THE MMS SYSTEM CONSIDERING THE COMMUNICATION

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

lookup 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.

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 three processors (with two voltage modes [(1.8 V, −0.1 V) and (1.0 V, −0.6 V)], communicating over a repeater-based shared bus. At the nomimal voltages, the communication accounts for 15% of the total energy consumption. Table V shows the resulting total energy consumptions for six different situations. The first column denotes the used voltage selection technique 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 μJ 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 fourth rows present the results of systems in which the bus remains unscaled while the processors are either $V_{\rm dd}$ or $V_{\rm dd}$ and $V_{\rm bs}$ 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 [29]. 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 1.723μJ, which is 24% below the unscaled system.

The MMS system is mapped on four 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.

XI. CONCLUSION

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 [39] and [40].

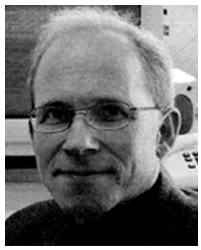
REFERENCES

[1] T. Ishihara and H. Yasuura, "Voltage scheduling problem for dynamically variable voltage processors," in *Proc. Int. Symp. Low Power Electronics and Design*, 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. Int. Conf. Comput.-Aided Des.*, 2002, pp. 721–725.
[3] F. Yao, A. Demers, and S. Shenker, "A scheduling model for reduced CPU energy," in *Proc. IEEE Symp. Foundations Comput. Sci*, 1995, pp. 374–382.
[4] C. Kim and K. Roy, "Dynamic Vth scaling scheme for active leakage power reduction," in *Proc. Design, Autom. Test Eur. Conf.*, 2002, pp. 163–167.
[5] S. Borkar, "Design challenges of technology scaling," *IEEE Micro*, vol. 19, no. 4, pp. 23–29, Jul. 1999.
[6] W. Kwon and T. Kim, "Optimal voltage allocation techniques for dynamically variable voltage processors," *ACM Trans. Embed. Comput. Syst.*, vol. 4, pp. 211–230, Feb. 2005.
[7] F. Gruian and K. Kuchcinski, "LEneS: Task scheduling for low-energy systems using variable supply voltage processors," in *Proc. ASP-DAC*, 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*, 2003, pp. 369–375.
[9] M. Schmitz and B. M. Al-Hashimi, "Considering power variations of DVS processing elements for energy minimization in distributed systems," in *Proc. Int. Symp. Syst. Synthesis*, 2001, pp. 250–255.
[10] Y. Zhang, X. Hu, and D. Chen, "Task scheduling and voltage selection for energy minimization," in *Proc. Des. Autom. Conf.*, 2002, pp. 183–188.
[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*, 2002, pp. 382–387.
[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 Syst. Symp.*, 1998, pp. 178–187.
[13] B. Mochocki, X. Hu, and G. Quan, "A realistic variable voltage scheduling model for real-time applications," in *Proc. Int. Conf. Comput.-Aided Des.*, 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*, 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. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 24, no. 7, pp. 1030–1041, Jul. 2005.
[16] R. G. R. Jejurikar, "Dynamic slack reclamation with procrastination scheduling in real-time embedded systems," in *Proc. Des. Autom. Conf.*, 2005, pp. 111–116.
[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, Jul. 2001.
[18] D. Sylvester and K. Keutzer, "Impact of small process geometries on microarchitectures in systems on a chip," *Proc. IEEE*, vol. 89, no. 4, pp. 467–489, Apr. 2001.
[19] Y. Ismail and E. Friedman, "Repeater insertion in RLC lines for minimum propagation delay," in *Proc. ISCAS*, 1999, pp. 404–407.
[20] P. Kapur, G. Chandra, and K. Saraswat, "Power estimation in global interconnects and its reduction using a novel repeater optimization methodology," in *Proc. DAC*, 2002, pp. 461–466.
[21] L. Shang, L. Peh, and N. Jha, "Power-efficient interconnection networks: Dynamic voltage scaling with links," *Comp. Arch. Lett.*, 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 processors," in *Proc. CODES*, 2002, pp. 169–174.
[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*, 1998, pp. 861–867.
[25] C.-T. Hsieh and M. Pedram, "Architectural energy optimization by bus splitting," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 21, no. 4, pp. 408–414, Apr. 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 Hardw./Softw. Co-Design (CODES)*, 1999, pp. 152–156.
[27] G. Varatkar and R. Marculescu, "Communication-aware task scheduling and voltage selection for total system energy minimization," in *Proc. Int. Conf. Comput.-Aided Des.*, 2003, pp. 510–517.
[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. Des., Autom. Test Eur. Conf.*, 2004, pp. 518–523.
[29] ——, "Simultaneous communication and processor voltage scaling for dynamic and leakage energy reduction in time-constrained systems," in *Proc. Int. Conf. Comput.-Aided Des.*, 2004, pp. 362–369.
[30] ——, "Energy optimization of multiprocessor systems on chip by voltage selection," Dept. Comput. Inf. Sci., Linköping Univ., Linköping, Sweden, 2007, Tech. Rep..
[31] ——, "Quasi-static voltage scaling for energy minimization with time constraints," in *Proc. DATE*, 2005, pp. 514–519.
[32] A. P. Chandrakasan and R. W. Brodersen, *Low Power Digital CMOS Design*. Norwell, MA: Kluwer, 1995.
[33] Intel, Santa Clara, CA, " Intel XScale Core, Developer's Manual," 2000.
[34] AMD, Sunnyvale, CA, "Mobile AMD Athlon 4, Processor Model 6 CPGA Data Sheet," Tech. Rep. 24319 Rev E, 2000.
[35] Y. Nesterov and A. Nemirovskii, "Interior-point polynomial algorithms in convex programming," in *Studies in Applied Mathematics*. Philadelphia, PA: SIAM, 1994.
[36] P. Caputa and C. Svensson, "Low-power, low-latency global interconnects," in *Proc. IEEE ASIC/SOC*, 2002, pp. 394–398.
[37] M. Schmitz, B. Al Hashimi, and P. Eles, *System-Level Design Techniques for Energy-Efficient Embedded Systems*. Norwell, MA: Kluwer, 2004.
[38] J. Hu and R. Marculescu, "Energy-aware mapping for tile-based NoC architectures under performance constraints," in *Proc. ASPDAC*, 2003, pp. 233–239.
[39] 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 MP-SoCs," in *Proc. Int. Symp. Syst.-on-Chip*, 2007, pp. 1–4.
[40] M. Schmitz, B. Al-Hashimi, and P. Eles, "Cosynthesis of energy-efficient multimode embedded systems with consideration of mode-execution probabilities," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 24, no. 2, pp. 153–169, Feb. 2005.

Alexandru Andrei (S'03) received the M.S. degree in computer science from Politehnica University Timisoara, Timisoara, Romania, in 2001. He is currently pursuing the Ph.D. degree in computer engineering from Linkoping University, Linkoping, Sweden.

His research interestes include low-power design, real-time systems, and hardware-software codesign.



Petru Eles (M'99) received the Ph.D. degree in computer science from the Politehnica University of Bucharest, Bucharest, Romania, in 1993.

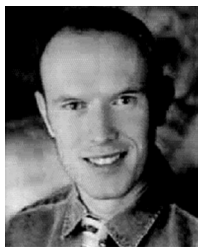
He is currently a Professor with the Department of Computer and Information Science at Linkoping University, Linkoping, Sweden. His research interests include embedded systems design, hardware-software codesign, real-time systems, system specification and testing, and CAD for digital systems. He has published extensively in these areas and coauthored several books.

Dr. Petru Eles is an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS and of the *IEE Proceeding—Computers and Digital Techniques*.

Zebo Peng (M'91–SM'02) received the B.Sc. degree in computer engineering from the South China Institute of Technology, Guangzhou, China, in 1982, and the Ph.D. degree in computer science from Linkoping University, Linkoping, Sweden, in 1985 and 1987, respectively.

Currently, he is Professor of Computer Systems and Director of the Embedded Systems Laboratory at the Department of Computer Science, Linkoping University. His research interests include design and test of embedded systems, design for testability, hardware/software co-design, and real-time systems. He has published 200 technical papers in these areas and co-authored several books.

Dr. Peng serves currently as the Chair of the IEEE European Test Technology Technical Council (ETTTC).



Marcus T. Schmitz received the diploma degree in electrical engineering from the University of Applied Science Koblenz, Koblenz, Germany, in 1999, and the Ph.D. degree in electronics from the University of Southampton, Southampton, U.K., in 2003.

He joined Robert Bosch GmbH, Stuttgart, Germany, where he is currently involved in the design of electronic engine control units. His research interests include system-level co-design, application-driven design methodologies, energy-efficient system design, and reconfigurable architectures.

Bashir M. Al-Hashimi (SM'01) received the B.Sc. degree (with first-class classification) in electrical and electronics engineering from the University of Bath, Bath, U.K., in 1984 and the Ph.D. degree from York University, York, U.K., in 1989.

In 1999, he joined the School of Electronics and Computer Science, Southampton University, Southampton, U.K., where he is currently a Professor of Computer Engineering. He has published over 180 technical papers and co-authored several books. His current research and teaching interests include low-power system-level design, system-on-chip test, and VLSI CAD.

Prof. Al-Hashimi is a Fellow of the Institution of Electrical Engineers (IEE), U.K. He is the Editor-in-Chief of the *IEE Proceedings—Computers and Digital Techniques*.