Temperature-Aware Idle Time Distribution for Energy Optimization with Dynamic Voltage Scaling

Min Bao
Linköping University
Sweden
minba@ida.liu.se

Alexandru Andrei
Ericsson, Linköping
Sweden
alexandru.andrei@ericsson.com

Petru Eles
Linköping University
Sweden
petel@ida.liu.se

Zebo Peng
Linköping University
Sweden
zpe@ida.liu.se

***Abstract*—With new technologies, temperature has become a major issue to be considered at system level design. In this paper we propose a temperature aware idle time distribution technique for energy optimization with dynamic voltage scaling (DVS). A temperature analysis approach is also proposed which is accurate and, yet, sufficiently fast to be used inside the optimization loop for idle time distribution and voltage selection.**

I. Introduction

Technology scaling and ever increasing demand for performance have resulted in high power densities in current circuits, which also lead to increased chip temperature. At the same time, the amount of leakage energy consumed can reach levels up to 70% of the total energy consumption [1]. Due to the strong dependence of leakage on temperature, growing temperature leads to an increase in leakage power and, consequently, energy, which, again, produces higher temperature. Thus, temperature is an important parameter to be taken into consideration during system level design.

At system level, dynamic voltage selection (DVS) [2] [3] is one of the preferred approaches for reducing the overall energy consumption. This technique exploits the available slack times to achieve energy efficiency by reducing the supply voltage and frequency such that the execution of tasks is stretched within their deadline. However, very often, not all available slack should or can be exploited and certain amount of slack may still exist after DVS. An obvious situation is when the lowest supply voltage is such that, even if selected, a certain slack is left until the deadline. Another reason is the existence of a critical voltage [4]. To achieve optimal energy efficiency, DVS would not execute a task at a voltage lower than the critical one, since, otherwise, the additional static energy consumed due to the longer execution time is larger than the energy saving due to the lowered voltage. During the available slack interval, the processor will be switched to a low power state.

Due to the dependence between leakage power and temperature, different distributions of idle time will lead to different temperature distributions and, consequentially, energy consumption. However, none of the previous DVS approaches has considered this issue. The closest work that considers idle time distribution (ITD) is [5], where an approach to distribute idle time with a given constant supply voltage is proposed. However, their approach only works for applications consisting of a single task, and cannot optimize the distribution of idle time among multiple tasks which also execute at different voltages. *In this paper, we address the issue of optimizing ITD globally among tasks, executing at different voltages, for energy minimization.*

Temperature aware system level design methods rely on the development of temperature modeling and analysis tools. Hotspot [6] is an architecture and system-level temperature model and simulator. The background theory of Hotspot is the duality between heat transfer and electrical phenomena [7]. Similar to Hotspot, the work in [8] proposes a temperature modeling approach, where dynamic adaptation of the resolution is performed, in order to speed up the thermal analysis.

However, temperature analysis time with approaches like the two mentioned above are too long to be affordable inside a temperature aware system level optimization loop. There has been some work on establishing fast system level temperature analysis techniques. They also build on the duality between heat transfer and electrical phenomena. Most of them are based on very restrictive assumptions in order to simplify the model. The work in [9] assumes that (1) no cooling layer is present, (2) there is no interdependency between leakage current and temperature, and (3) the whole application executes at constant voltage. The models in [10], [11] consider variable voltage levels but maintain the first two limitations above. The most general analytical model is proposed in [12] which considers cooling layers as well as the dependency between leakage and temperature. However, this approach is limited to the case of a unique voltage level throughout the application. *In order to support the idle time distribution technique with DVS, proposed in this paper, we introduce, as a second contribution of this work, a fast and accurate temperature analysis technique that eliminates all three limitations mentioned above.*

II. Preliminaries

A. Power and Application Model

For dynamic power we use the following equation [13]: $P_d = C_{eff} * f * V^2$, where C_{eff}, V, and f denote the effective switched capacitance, supply voltage, and frequency, respectively.

The leakage power is expressed as follows [14]:

$$P_{leak} = I_{sr} * T^2 * e^{\frac{\beta * V + \gamma}{T}} * V \tag{1}$$

where I_{sr} is the reference leakage current at reference temperature, T is the current temperature. β and γ are technology dependent coefficients. In Section IV-B, in particular, we will use a piece-wise linear approximation of this model as proposed, for example, in [15]. According to it, the working temperature range (T_a, T_{max})[1] is divided into several sub-ranges. The leakage power inside each sub-range (T_i, T_{i+1}) is modeled by a linear function: $P_i = K_i * T + B_i$, where K_i and B_i are constants characteristic to each interval.

The application is captured as a set of task graphs $G(\Pi, \Gamma)$. Nodes $\tau \in \Pi$ represent computational tasks, while edges indicate data dependencies between tasks. Each task is characterized by the following parameters: the maximum number of clock cycles to be executed, a deadline, and the average switched capacitance.

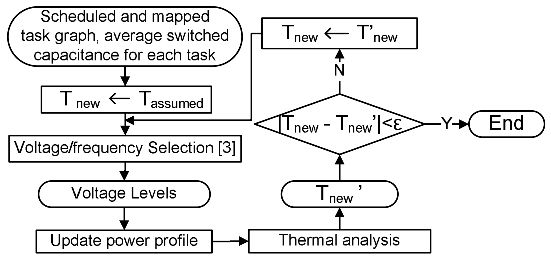
The application is mapped and scheduled on a processor that has two power states: active and idle. In active state the processor can operate at several discrete supply voltage levels. When the processor does not execute any task, it can be put to the idle state, consuming a very small amount of leakage power. Switching the processor between the idle and active state as well as between different voltage levels incurs time and energy overheads.

B. Temperature Aware DVS

In [2] we have presented a DVS approach which, given a mapped and scheduled application, calculates the voltage levels for each task, such that the total energy consumption is minimized and deadlines are satisfied. Another input to the algorithm is the dynamic power profile of the application, which is captured by the average switched capacitance of each task. This information is used for calculating the dynamic energy consumed by the task executed at certain supply voltage levels, according to the dynamic power model in Section II-A. The leakage energy is calculated using Eq. 1.

[1] T_a and T_{max} are the ambient and the maximal working temperature of the chip.

Based on the approach in [2], in [16] we have proposed a temperature aware DVS technique illustrated in Fig. 1. The approach starts from an initial assumed temperature, at which the processor is guessed to run. The voltage selection algorithm will determine, for each task, the voltage levels such that energy consumption is minimized. Based on the determined voltage (and the switched capacitances known for each task) the power profiles are calculated, the thermal analysis is performed, and the processor temperature profile is determined in steady state. This new temperature information is now used again for voltage selection and the process is repeated until the temperature converges. As shown in [16], in most of the cases, convergence is reached in less than 5 iterations.



Scheduled and mapped task graph, average switched capacitance for each task
$T_{new} \leftarrow T_{assumed}$
Voltage/frequency Selection [3]
Voltage Levels
Update power profile
Thermal analysis
T_{new}'
$|T_{new} - T_{new}'|<\varepsilon$
Y
End
N
$T_{new} \leftarrow T'_{new}$

Fig. 1. Temperature Aware DVS

The above approach, however, ignores the influence of the ITD on the energy consumption. The placement of idle time is arbitrary, e.g., it is placed after finishing all tasks, ignoring the interdependency between idle slots placement and temperature. On the other hand, in order to perform an efficient ITD, dynamic temperature analysis is needed. Such an analysis generates the temperature variation curve, as opposed to simple static steady state analysis which only produces an estimated steady state temperature value. Dynamic temperature analysis is much more time consuming than the static steady state one. The DVS technique in Fig. 1 uses Hotspot to produce the static steady state temperature. Once ITD is taken into consideration, using Hotspot for dynamic temperature analysis is not affordable due to the extremely long execution time.

The rest of the paper is devoted to the two problems identified above: (1) the need for a fast but accurate dynamic temperature analysis and (2) an efficient idle time distribution technique.

III. Motivational Example

Let us consider an application consisting of 5 tasks which share a global deadline of 88.6ms. The worst case workload (in clock cycles) and C_{eff} are given in Table I. The supply voltage (V) and frequency $(Freq.)$, as calculated by the temperature aware DVS (Section II-B), are also presented in Table I, as well as the corresponding task execution time $(time)$.

TABLE I
Motivational Example: Application

	$Workload$	$C_{eff}(f)$	$V(V)$	$Freq.(MHZ)$	$time(ms)$
τ_1	1.10e7	2.26e-08	0.6	132	9.8
τ_2	1.59e7	3.09e-08	0.6	132	14.0
τ_3	1.27e7	2.85e-08	0.6	132	11.2
τ_4	1.46e7	4.66e-08	0.55	108	16.3
τ_5	1.46e7	3.49e-08	0.6	132	12.9

Based on the voltage and frequency assignment, there is an idle time (t_{idle}) of 24.4ms. Fig. 2 gives two different ways of distributing t_{idle}. The 1st distribution (1st ITD), as shown in Fig. 2a, places the whole t_{idle} after the last task, while the 2nd distribution (2nd ITD), in Fig. 2b, inserts an idle slot after each task.

For simplicity, in this example, we ignore both energy and time overhead due to switching between active and idle mode. The two different ITDs will lead to different temperature and leakage power profiles. The average working temperature Tw of each task as well

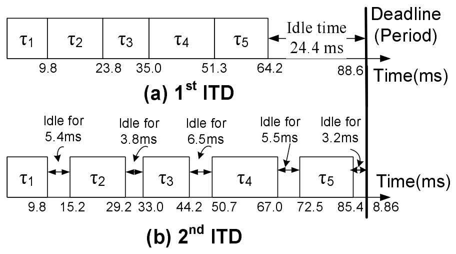


Fig. 2. Motivational Example: Idle Time Distribution

as the energy consumption are shown in Table II, where E_d, E_l and $E_{tot}(J)$ are the dynamic, leakage and total energy consumption respectively. Around 17% reduction of leakage energy consumption and 10% reduction of the total energy consumption are observed comparing the 2nd ITD with the 1st ITD.

TABLE II
Motivational Example: Energy Comparison

		$E_d(J)$	$E_l(J)$	$E_{tot}(J)$	Tw(°C).
1st ITD:	τ_1	0.090	0.102	0.192	58.3
	τ_2	0.176	0.213	0.390	75.2
	τ_3	0.130	0.222	0.352	87.9
	τ_4	0.206	0.319	0.525	95.5
	τ_5	0.183	0.351	0.534	104.5
$Tot.$		0.787	1.206	1.993	
2nd ITD:	τ_1	0.090	0.160	0.250	80.0
	τ_2	0.176	0.217	0.394	78.2
	τ_3	0.130	0.189	0.319	80.3
	τ_4	0.206	0.224	0.430	76.5
	τ_5	0.183	0.214	0.397	78.2
$Tot.$		0.787	1.003	1.790	

The energy reduction is due to the modified working temperature of the chip which has a strong impact on the leakage power. It is also important to mention that the table reflects the steady state (not the start-up mode), for which energy minimization is targeted. This means that the starting temperature for τ_1 is identical to the temperature at the end of the previous period.

IV. Temperature Analysis

According to the model in Section II-A, the processor executes the application periodically with each task executed at the voltage level calculated off line by the DVS algorithm. Thus, the processor temperature will, finally, converge to a steady state dynamic temperature curve (SSDTC) which repeats periodically.

A. Temperature Model

Thermal Circuit. In order to analyze the thermal behavior, we build an equivalent RC thermal circuit based on the physical parameters of the die and the package [7]. Due to the fact that the application period t_p can safely be considered significantly smaller than the RC time of the heat sink, which is usually in the order of minutes [12], the heat sink temperature stays constant after the state corresponding to the SSDTC is reached. We, hence, can ignore the thermal capacitance (not the thermal resistance!) of the heat sink and build the 2-RC thermal circuit shown in Fig. 3a. B_1 and B_2 represent the temperature node for the die and the heat spreader respectively. $P(t)$ stands for the processor power consumption as a function of time.

We obtain the values of R_1, R_2, C_1 and C_2 from an RC network similar to the one constructed in Hotspot [6]. R_1 is calculated as the sum of the thermal resistance of the die and the thermal interface material (TIM), and C_1 as the sum of the thermal capacitance of the die and the TIM. R_2 is the equivalent thermal resistance from the heat spreader to the ground through the heat sink, and C_2 the equivalent thermal capacitance of the heat spreader layer.

When the application period t_p is significantly smaller than the RC time of the heat spreader in the 2-RC thermal circuit, the heat

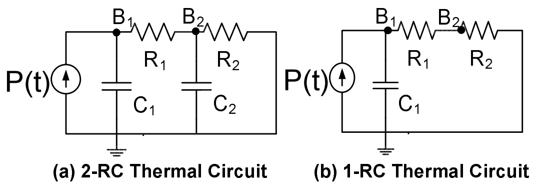


Fig. 3. Thermal Circuit

spreader temperature stays constant after SSDTC is reached. In this case, we can simplify the 2-RC to an 1-RC thermal circuit (Fig. 3b).

Temperature Equations. For the 2-RC thermal circuit in Fig. 3a, we can describe the temperatures of B_1 and B_2 as follows:

$$C_1 * \frac{dT^{die}}{dt} + \frac{T^{die} - T^{sp}}{R_1} = P(t) \quad (2)$$

$$C_2 * \frac{dT^{sp}}{dt} + \frac{T^{sp}}{R_2} = \frac{T^{die} - T^{sp}}{R_1} \quad (3)$$

where T^{die} and T^{sp} represent the temperature at B_1 and B_2.The power consumption $P(t)$ is the sum of the dynamic and leakage power, which are dependent on the supply voltage V and on T^{die}.

If, within a time interval, the power consumption stays constant P, the temperature at the beginning and end of the time interval can be expressed as follows, by solving Eq. 2 and Eq. 3:

$$T_e^{die} = a_1 * T_b^{die} + b_1 * T_b^{sp} + c_1 \quad (4)$$

$$T_e^{sp} = a_2 * T_b^{die} + b_2 * T_b^{sp} + c_2 \quad (5)$$

T_b^{die} and T_b^{sp} are the temperature of B_1 and B_2 at the beginning, while T_e^{die} and T_e^{sp} are the temperature at the end of the time interval. a_1, a_2, b_1, b_2, c_1 and c_2 are constant coefficients determined by R_1, R_2, C_1, C_2 and P.

B. SSDTC Estimation

As an input to the SSDTC calculation we have the voltage levels, calculated by the DVS algorithm, and a given idle time distribution, as illustrated in Fig. 4a.

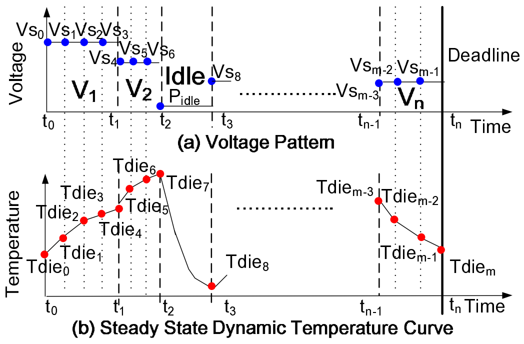


Fig. 4. Temperature Analysis

When the processor is working in active state, the leakage power consumption varies with the working temperature of the processor. In Fig. 4a, we divide the execution interval of each active state step into several sub-intervals. The total number of sub-intervals is denoted as m. Each sub-interval is short enough such that the temperature variation is small and the leakage power can be treated as constant inside the sub-interval.

P_i is the power consumption for each sub-interval i $(1 \leq i \leq m)$. When the processor is in active state during the i_{th} sub-interval, P_i is computed by Eq. 6, where Vs_{i-1} and T_{i-1}^{die} are the supply voltage and processor temperature at the start of the i_{th} sub-interval. $P_d(Vs_{i-1})$ represents the dynamic power consumption while $P_l(T_{i-1}^{die}, Vs_{i-1})$ represents the leakage power consumption based on the piece-wise linear leakage model discussed in Section II-A. When the processor is in idle state during the ith sub-interval, the power consumption $P_i = P_{idle}$.

$$P_i = P_d(Vs_{i-1}) + P_l(T_{i-1}^{die}, Vs_{i-1}) \quad (6)$$

As shown in Fig. 4b, we construct the SSDTC by calculating the temperature values T_0^{die} to T_m^{die}. The relationship between the start and end temperature of each sub-interval can be described by applying Eq. 4 and Eq. 5 to all sub-intervals. Thus, we can establish a linear system with $2 * m$ equations as follows:

$$T_1^{die} = a_{11} * T_0^{die} + b_{11} * T_0^{sp} + c_{11} \quad (7)$$

$$T_1^{sp} = a_{12} * T_0^{die} + b_{12} * T_0^{sp} + c_{12} \quad (8)$$

$$\cdots\cdots\cdots$$

$$T_m^{die} = a_{m1} * T_{m-1}^{die} + b_{m1} * T_{m-1}^{sp} + c_{m1} \quad (9)$$

$$T_m^{sp} = a_{m2} * T_{m-1}^{die} + b_{m2} * T_{m-1}^{sp} + c_{m2} \quad (10)$$

T_i^{die} and T_i^{sp} are the temperature of the processor and heat spreader at the beginning of the $i + 1_{th}$ sub-interval. Due to periodicity, when dynamic steady state is reached, the processor and heat spreader temperature at the beginning of the period should be equal with the temperature values at the end of the previous period:

$$T_0^{die} = T_m^{die};\ T_0^{sp} = T_m^{sp} \quad (11)$$

Solving the above linear system, we get the values for T_0^{die} to T_m^{die} and, hence, obtain the corresponding SSDTC.

V. Idle Time Distribution

A. Problem Formulation

Let us consider a set of tasks $(\tau_1, \tau_2, \ldots, \tau_n)$ executed in the order τ_1, τ_2, …, τ_n. For each task τ_i, its deadline dl_i, the supply voltage level V_i at which the task is executed (calculated by the temperature aware DVS algorithm) and the corresponding worst case execution time te_i are given. The total idle time of the processor t_{idle} is $dl_n - \sum_{i=1}^{n}(te_i)$. During t_{idle}, the processor can be switched to idle mode consuming the power P_{idle}. The time and energy overhead for switching the processor to and from the idle state are t_o and E_o respectively. Idle slots can be placed after the execution of any task. The length of an idle slot i after task τ_i is denoted as t_i, and the sum of all idle slots $\sum_{i=1}^{n}(t_i)$ should be equal with the total available idle time t_{idle}[2]. The total energy consumption E_{tot} is computed by Eq. 12:

$$E_{tot} = \sum_{i=1}^{n}(P_{d_i}(V_i) * te_i) + \sum_{i=1}^{n} El_i + \sum_{i=1}^{n}(E_o * x_i) + E_I \quad (12)$$

where $\sum_{i=1}^{n}(P_{d_i}(V_i) * te_i)$ and $\sum_{i=1}^{n} El_i$ are the total dynamic and leakage energy consumption during task execution. $\sum_{i=1}^{n}(E_o * x_i)$ is the total energy overhead when the processor is switched to/from idle state, where x_i is a binary variable indicating whether task τ_i is followed $(x_i = 1\ )$ or not $(x_i = 0\ )$ by an idle slot. E_I is the total energy consumption during the idle time t_{idle}: $E_I = P_{idle} * t_{idle}$.

With the given supply voltage V_i for each task τ_i, the total dynamic energy consumption $\sum_{i=1}^{n}(P_{d_i}(V_i) * te_i)$ in Eq. 12 and the the energy consumption during the idle time E_I are fixed and do not depend on the idle time distribution. Our ITD problem is then formulated as follows: with given amount of t_{idle}, determine the values $t_i (\forall 1 \leq i \leq n)$, with the constraint that $t_{idle} = \sum_{i=1}^{n}(t_i)$, such that the following sum is minimized:

$$\sum_{i=1}^{n} El_i + \sum_{i=1}^{n}(E_o * x_i) \quad (13)$$

In Section V-B we first introduce an idle time distribution approach ignoring the overheads E_o and t_o. This approach will, then, be used in Section V-C where we present our general idle time distribution technique with overheads.

B. ITD without overhead (ITDNOH)

Let us first consider a particular case: the execution time te_i of each task τ_i is long enough such that the processor and heat spreader reach

[2]The time overhead t_o is included in the idle time slot t_i

a steady state temperature Ts_i^{die} and Ts_i^{sp} when task τ_i finished (the temperature does not further change if the task would continue). As opposed to the temperature at the beginning of the task τ_i, which depends on the length of the idle period, Ts_i^{die} and Ts_i^{sp} are independent of the idle time distribution and are calculated as follows, based on our 2-RC thermal circuit (Fig. 3):

$$\begin{aligned} Ts_i^{die} &= (Pd_i(V_i) + Pl_i(V_i, Ts_i^{die})) * (R_1 + R_2) && (14) \\ Ts_i^{sp} &= (Pd_i(V_i) + Pl_i(V_i, Ts_i^{die})) * R_2 && (15) \end{aligned}$$

where R_1 and R_2 are the same as introduced in Section IV-A. $Pl_i(V_i, Ts_i^{die})$ is the leakage power at the temperature Ts_i^{die}. By solving the above equations we obtain Ts_i^{die} and Ts_i^{sp}.

Since, in this section, we ignore the overheads ($E_o = t_o = 0$), from Eq. 13, it results that the cost to be minimized is $\sum_{i=1}^{n} El_i$, which is the total leakage energy consumed during task execution.

Assuming that the execution interval of task τ_i is divided into a number of $q_i - 1$ sub-intervals, the total leakage energy consumption of τ_i is the sum of the leakage energy of all sub-intervals:

$$El_i = \sum_{j=1}^{q_i-1} (P_{l_{ij}}(V_{ij}, \frac{T_{ij}^{die} + T_{i(j+1)}^{die}}{2}) * t_{ij}^{sub}) \qquad (16)$$

where T_{ij}^{die}, $T_{i(j+1)}^{die}$ and t_{ij}^{sub} represent the processor temperature at the beginning and end of the j_{th} sub-interval and the length of this sub-interval, respectively. The leakage model in Eq. 1 is used to compute the leakage power in each sub-interval.

We can formulate our ITDNOH problem for this scenario as shown in Eq. 17-Eq. 29 where the objective function to be minimized is the total leakage energy for all tasks (Eq. 17):

$$\sum_{i=1}^{n} (\sum_{j=1}^{q_i-1} (t_{ij}^{sub} * P_{l_{ij}}(V_{ij}, \frac{T_{ij}^{die} + T_{i(j+1)}^{die}}{2}))) \qquad (17)$$

Subject to:

$$\begin{aligned} t_{idle} &= \sum_{i=1}^{n} (t_i) && (18) \\ t_i &\geq 0 (1 \leq i \leq n) && (19) \\ dl_i &\geq \sum_{j=1}^{i-1} t_j + \sum_{w=1}^{i} \sum_{j=1}^{q_i-1} t_{wj}^{sub} && (20) \\ T_{iq_i}^{die} &= Ts_i^{die} (1 \leq i \leq n) \; see \; Eq. \; 14 && (21) \\ T_{iq_i}^{sp} &= Ts_i^{sp} (1 \leq i \leq n) \; see \; Eq. \; 15 && (22) \\ T_{i(j+1)}^{die} &= a1_{ij} * T_{ij}^{die} + b1_{ij} * T_{ij}^{sp} + c1_{ij} && (23) \\ T_{i(j+1)}^{sp} &= a2_{ij} * T_{ij}^{die} + b2_{ij} * T_{ij}^{sp} + c2_{ij} && (24) \\ &(\quad 1 \leq i \leq n; 1 \leq j \leq q_i - 2) && \\ T_{(i+1)1}^{die} &\geq TIs + (T_{iq_i}^{die} - TIs) * exp(\frac{-t_i}{R_g * C_1}) && (25) \\ &(\quad 1 \leq i \leq n-1) && \\ T_{(i+1)1}^{sp} &= T_{(i)q_i}^{sp} \; (1 \leq i \leq n-1) && (26) \\ T_{11}^{die} &\geq TIs + (T_{nq_n}^{die} - TIs) * exp(\frac{-t_n}{R_g * C_1}) && (27) \\ T_{11}^{sp} &= T_{nq_n}^{sp} && (28) \\ TIs &= P_{idle} * R_g && (29) \end{aligned}$$

The optimization variables to be calculated are the idle slot lengths $t_i (\forall 1 \leq i \leq n)$. $T_{iq_i}^{die}$ and $T_{iq_i}^{sp}$ are the processor and heat spreader temperature at the end of execution of task τ_i and they are equal to the steady state temperature Ts_i^{die} and Ts_i^{sp} respectively (Eq. 21, Eq. 22). T_{ij}^{die} and $T_{i(j+1)}^{die}$ are the processor temperature at the beginning and end of j_{th} sub-interval in the execution of task τ_i and are given by Eq. 23 which is similar to Eq. 7 and Eq. 9 in Section IV-B. Eq. 24 describes the same relationship for the heat spreader temperature. $T_{(i+1)1}^{die}$ and $T_{(i+1)1}^{sp}$ are the processor and heat spreader temperature at the start of task τ_{i+1}, and are dependent on the finishing temperature of the previous task τ_i and the idle slot t_i placed after τ_i. If we assume that all idle slots t_i are significantly shorter than the RC time of the heat spreader, then we can describe the processor temperature behavior during the idle slot i by Eq. 25 and Eq. 27, based on the 1-RC thermal circuit described in Section IV-A. TIs is the steady state temperature that the processor would reach if P_{idle} would be consumed for a sufficiently long time and is calculated according to Eq. 29. R_g is the sum of the two thermal resistances R_1 and R_2 in Fig. 3b. Under the same assumption, the heat spreader temperature stays constant during the idle slot as shown in Eq. 26 and Eq. 28[3]. Eq. 25 and Eq. 26 calculate the processor and heat spreader temperature at the end of the idle slot following task τ_i and, implicitly, the staring temperature of τ_{i+1}. Eq. 27 and Eq. 28 compute the temperature at the start of task τ_1, taking into consideration that this task starts after the idle period following task τ_n (the task set is executed periodically). The above formulation is a convex nonlinear problem, and can be solved efficiently in polynomial time [17].

ITDNOH Approach. For the situation when tasks can have arbitrary execution time, the temperature of the processor and heat spreader at the end of task τ_i may not reach the steady state temperature. Thus, the values of $T_{iq_i}^{die}$ and $T_{iq_i}^{sp}$ (Eq. 21, Eq. 22) are no longer constants but are dependent on the idle time distribution. This makes the above formulation become a non-convex programming problem which is very time consuming to solve. In order to solve the problem efficiently we have developed an iterative heuristic outlined in Fig. 5.

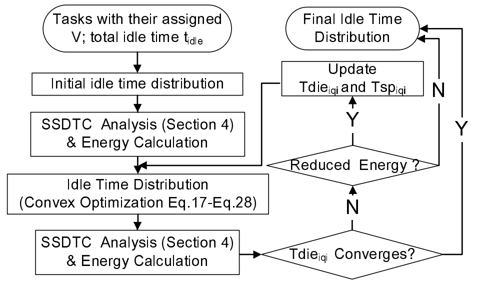

Tasks with their assigned V; total idle time t_{idle}
Final Idle Time Distribution
Initial idle time distribution
Update $Tdie_{iqi}$ and Tsp_{iqi}
N
Y
Y
SSDTC Analysis (Section 4) & Energy Calculation
Reduced Energy ?
Idle Time Distribution (Convex Optimization Eq.17-Eq.28)
N
SSDTC Analysis (Section 4) & Energy Calculation
$Tdie_{iqi}$ Converges?

Fig. 5. ITDNOH Heuristic

The heuristic starts with an arbitrary initial ITD, for example, that the entire idle time t_{idle} is placed after the last task τ_n. Assuming this ITD and the given voltage levels, steady state dynamic temperature analysis is performed, as described in Section IV-B. Given the obtained SSDTC and the voltage levels, the total energy consumption E_{tot} corresponding to the assumed ITD is calculated. From the SSDTC we can also extract the final temperature $T_{iq_i}^{die}$ and $T_{iq_i}^{sp}$ for each task τ_i. Assuming this $T_{iq_i}^{die}$ and $T_{iq_i}^{sp}$ to be the final temperature in Eq. 21 and Eq. 22, we can calculate the idle time t_i using the convex optimization formulated in Eq. 17-Eq. 29.

From the new ITD resulted after the optimization, we calculate a new SSDTC which provides new temperatures $T_{iq_i}^{die}$ and $T_{iq_i}^{sp}$ at the end of each task τ_i. The new total energy consumption E_{tot}, corresponding to the updated ITD, is also calculated. The process is continued assuming the new end temperatures in Eq. 21 and Eq. 22 and the convex optimization produces a new ITD.

The iteration process outlined above stops when the temperature $T_{iq_i}^{die}$ converges (i.e. $|T_{iq_i}^{die^{new}} - T_{iq_i}^{die^{old}}| < \varepsilon, \forall i \; 1 \leq i \leq n$). However, it can happen that, after a certain point, additional iterations do not significantly improve the ITD. Therefore, even if convergence has not yet been reached, the optimization is stopped if no significant

[3]Idle periods are supposed to be short. If, exceptionally, they are not significantly shorter than the heat spreader RC time, we use the 2-RC circuit to model the temperature during the idle period in Eq. 25—Eq. 28. This will not affect the convexity of the formulation.

energy reduction has been achieved ($(E_{tot}^{old} - E_{tot}^{new})/E_{tot}^{old} < \varepsilon'$). Our experiments have shown that maximum 5 iterations are needed with $\varepsilon = 0.5°$ and $\varepsilon' = 0.1\%$.

C. ITD with overhead (ITDOH)

The approaches presented in section V-B are based on the assumption that time and energy overhead t_o and E_o are zero, which is not the case in reality. If we consider the restricted case with tasks such that the end temperature does not depend on the idle time distribution, the problem can be formulated similar to Eq. 17-Eq. 29, with the main difference that the total energy to be minimized is Eq. 13. Based on this formulation, we could solve the ITDOH for the general case, when tasks can have arbitrary execution time, similarly with the approach described in Fig. 5. However, the formulation with the objective function Eq. 13, due to the binary variable x_i, is a mixed integer convex programing problem which is very time consuming to solve. We, hence, propose an ITDOH heuristic based on the ITDNOH approach in Fig. 5.

Our ITDOH heuristic comprises two steps. In the first step an optimization of the idle time distribution is performed by eliminating idle intervals whose length is shorter than a certain threshold limit. In the second step, the ITD is further refined in order to improve energy efficiency.

A lower bound t^{min} on the length t_i of an idle slot can be determined by considering the following two limitations:

1) No idle slot is allowed to be shorter than t_o, the total time needed to switch to/from the idle state.
2) The energy overhead due to switching should be compensated by the gain due to putting the processor into the idle state. The energy gain for an idle interval t_i is $(P_l - P_{idle}) * t_i$ where P_l is the leakage power consumption in active state. Thus, in order for the overhead to be compensated, we need $E_o < (P_l - P_{idle}) * t_i$. However, P_l depends on the processor temperature and, thus, the threshold length of an idle slot is not a given constant. Nevertheless, this threshold will be always longer than $E_o/(P_l^{max} - P_{idle})$, where P_l^{max} is the leakage power at the maximum temperature at which the processor is allowed to run.

In conclusion, for the first step of the ITDOH heuristic, illustrated in Fig. 6a, we consider: $t^{min} = max(t_o, E_o/(P_l^{max} - P_{idle}))$.

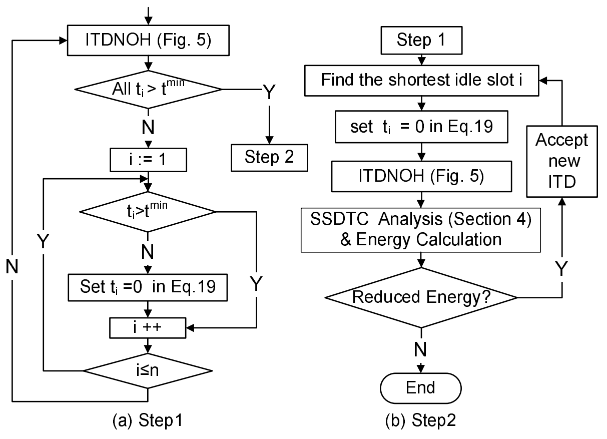


Fig. 6. ITDOH Heuristic

The basic idea of the first step is that no idle slot is allowed to be shorter than t^{min}. Thus, after running ITDNOH, the obtained ITD is checked slot by slot. If a slot length t_i is shorter than t^{min}, this slot will be forced to disappear. In order to achieve this, the particular constraint in Eq. 19, corresponding to slot i, is changed from $t_i \geq 0$ to $t_i = 0$. After all slots have been visited and Eq. 19 updated, ITDNOH is performed again. The obtained ITD is such that all slots which in the previous iteration have been found shorter than t^{min} have disappeared. The process is repeated until no slot shorter than t^{min} has been identified.

After step1, we still can be left with slots that are too short from the energy efficiency point of view. There are two reasons for this:

1) Due to the fact that the processor is running at a temperature lower than the maximum, worst case, one, it can happen that $E_o > (P_l - P_{idle}) * t_i$.
2) Even if $E_o < (P_l - P_{idle}) * t_i$, which means that there exists a certain energy reduction due to the idle slot, energy efficiency can, possibly, be improved by eliminating the slot. Eliminating a slot means, implicitly, distributing the corresponding amount of idle time between other slots, since the total idle time, t_{idle}, is constant for the given voltage levels per task.

In the second step, illustrated in Fig. 6b, we start from the shortest idle slot and consider to eliminate it (by setting the corresponding constraint in Eq. 19). If the ITD obtained after applying ITDNOH is more energy efficient, the new ITD is accepted. The process is continued, as long as, by eliminating a slot, the total amount of energy consumed is reduced.

VI. Experimental Results

The platform parameters used in our experiments are based on values taken from [18], [19] and [20]. We consider platforms with a die area of 6*6, 8*8 and 10*10mm^2. The heat spreader area is five times the die area and the heat sink area is between 1.3 and 1.4 times the area of the heat spreader. The thickness of the die and heat spreader are 0.5mm, and 2mm respectively. The thickness of the heat sink is between 10mm and 20mm. The coefficients corresponding to the power model in Section II-A are based on [13] [3]. For the SSDTC calculation (Section IV-B) we have considered a piece-wise linear leakage model with 3 segments, as recommended in [15].

The first set of experiments evaluate the accuracy of our proposed temperature analysis approach. We randomly generated 500 periodic voltage patterns corresponding to applications with periods in the range between 5ms and 100ms. For each application, considering the coefficients and platform parameters outlined above, we have computed the SSDTC using the approach proposed in Section IV-B and by using Hotspot simulation. For each pair of temperature curves obtained, we calculated the maximum deviation, as the largest temperature difference between any corresponding pairs of points (in absolute value), as well as the average deviation. Fig. 7 illustrates the results for different application periods. For applications with a period of 50ms, for example, there is no single case with a maximum deviation larger than $2.1°C$, and the average deviation is $1°C$. Over all 500 applications, the average and maximum deviation are $0.8°C$ and $3.8°C$ respectively. We can observe that the deviation increases with the increasing period of the application. This is due to the fact that, with larger periods, accuracy can be slightly affected by neglecting the thermal capacitance of the heat sink (see Section IV-A). We also compare the computation time of our

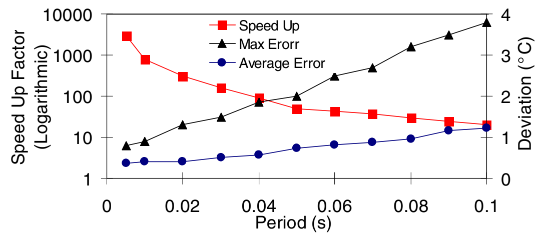


Fig. 7. SSDTC Estimation with Our Approach VS. Hotspot

SSDTC generation approach with the time needed by Hotspot. Fig. 7 illustrates the average speedup as the ratio of the two execution times. The speedup is between 3000 for periods of 5ms and 20 for 100ms periods. An increasing period leads to a larger linear system that has to be solved for SSDTC estimation (Section IV-B), which explains the shape of the speedup curve in Fig. 7.

Of course, the accuracy and speedup of our approach are also dependent on the length of the sub-interval considered for the temperature analysis (Section IV-B and Fig. 4). For the experiments throughout this paper, the length of the sub-interval is 2ms. This is based on the observation that reducing the length beyond this limit does not improve the accuracy significantly.

The next set of experiments aims to evaluate the efficiency of our ITDOH heuristic presented in Section V-C. We have randomly generated 1000 applications consisting of 2 to 100 tasks. The workload of each task is in the range [10^6,10^7] clock cycles. The applications are executed on platforms as discussed at the start of this section. For each application we have performed the following steps:

1) The temperature aware DVS algorithm described in Section II-B is run to determine the voltage levels for each task.
2) Based on the obtained voltage assignment, we compute t_{idle} as $t_p - \sum_{i=1}^{n}(te_i)$, where t_p is the deadline of the application, te_i is the execution time of task τ_i at the assigned voltage, and n is the number of tasks in the application. The ratio between t_{idle} and t_p is denoted as *idle time ratio*.
3) If $t_{idle} > 0$, the energy consumption E_1 of the application is calculated, with the assumption that t_{idle} is placed after the last task.
4) If $t_{idle} > 0$, we apply our ITDOH approach described in Section V-C. The corresponding energy consumption E_2 is calculated. The energy reduction is computed as $(E_1 - E_2)/E_1 * 100\%$.

The above four steps are performed, for each application, considering three different contexts with regard to the energy and time overheads E_o and t_o. The setting corresponding to $E_o = 0.5mJ$ and $t_o = 0.4ms$ is based on the values indicated in [21]. The two other settings assume a lower ($E_o = 0.25mJ, t_o = 0.2ms$) and higher ($E_o = 1mJ, t_o = 0.8ms$) overhead respectively. Fig. 8 shows the averaged energy reduction due to ITD, corresponding to different idle time ratios and overheads. The energy reduction achieved by ITD grows with the available amount of idle time and reaches 15% with an idle time ratio of 20-25%.

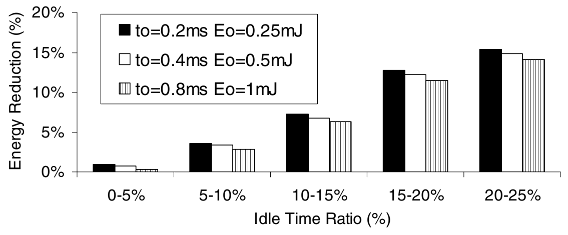


Fig. 8. Energy Reduction by ITDOH

We also evaluated the computation time for our ITDOH approach. Fig. 9 shows the results averaged for the test applications having $15\% \sim 25\%$ idle time ratio. The line marked with triangles shows the computation time of our ITDOH approach (the time consumed in step 4 above) as function of the number of tasks in the application. The computation time is within only 8 seconds for even very large applications. In Fig. 9 we also indicate the total execution time of all 4 steps, including both DVS and ITD.

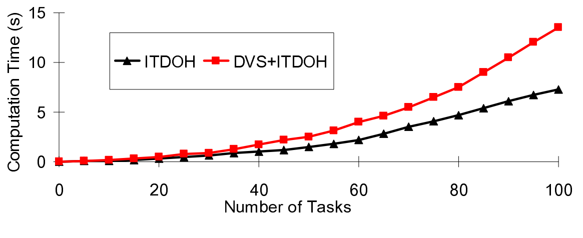


Fig. 9. Computation Time

We have also applied our ITDOH approach to a real life case, namely an MPEG2 decoder which consists of 34 tasks and is described in detail in [22]. We performed the fours steps described above, considering the same three settings for the overhead E_o and t_o. The energy reduction by applying our ITDOH approach is 12.2%, 11.8% and 10.6% respectively.

And, finally, a word on the integration between the two techniques, DVS and ITD. In the experiments described above we have first performed the temperature aware DVS and, using the obtained voltage levels and total idle time as an input, we have run our ITDOH algorithm. Another alternative is also possible, namely, to place the ITDOH algorithm inside the DVS optimization loop in Fig. 1, after the voltage/frequency selection step. Considering this alternative, we have performed experiments on the same applications and platforms as described above. The obtained energy reduction was, on average, $2\% \sim 3\%$ larger than by applying DVS and ITD in sequence. However, by placing ITD inside the loop, the total execution time becomes three times larger on average. This increased execution time (e.g. 45 seconds for 100 tasks), is, nevertheless, affordable if such an improved level of optimization is required.

VII. Conclusion

We have proposed an idle time distribution heuristic for energy minimization. In order to efficiently perform temperature analysis inside our optimization loop, we have also proposed a fast and accurate system level temperature analysis approach. Experiments show that our temperature analysis method achieves good accuracy with fast speed. The experiments also demonstrate that considerable energy reduction can be achieved by an efficient idle time distribution in the context of temperature aware DVS.

References

[1] "http://public.itrs.net., international technology roadmap for semiconductors."
[2] A. Andrei, P. Eles, and Z. Peng, "Energy optimization of multiprocessor systems on chip by voltage selection," *IEEE Trans. on Very Large Scale Integration Systems*, vol. 15, no. 3, p. 262C275, March 2007.
[3] Y. Liu, H. Yang, R. Dick, H. Wang, and L. Shang, "Thermal vs energy optimization for dvfs-enabled processors in embedded systems," *International Symposium on Quality Electronic Design*, pp. 204–209, Mar. 2007.
[4] R. Jejurikar, C. Pereira, and R. Gupta, "Leakage aware dynamic voltage scaling for realtime embedded systems," *Design automation Conference*, pp. 275 – 280, Jun. 2004.
[5] L. Yuan, S. Leventhal, and G. Qu, "Temperature-aware leakage minimization technique for real-time systems," *International Conference on Computer Aided Design*, pp. 761–764, 2006.
[6] W.Huang, S.Ghosh, S.Velusamy, K. Sankaranarayanan, K. Skadron, and M. Stan, "Hotspot: A compact thermal modeling methodology for early-stage vlsi design," *IEEE Trans. VLSI Systems*, vol. 14, no. 5, pp. 501–513, May 2006.
[7] A. Krum, *Thermal management. In The CRC Handbook of Thermal Engineering*. Boca Raton: F. Kreith, Ed. CRC Press, 2000.
[8] Y. Yang, Z. P. Gu, R. P. Dick, and L. Shang, "Isac: Integrated space and time adaptive chip-package thermal analysis," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, pp. 86–99, Jan. 2007.
[9] S. Wang and R. Bettatin, "Delay analysis in temp.-constrained hard real-time systems with general task arrivals," *Real-Time Systems Symposium*, pp. 323–334, 2006.
[10] R. Jayaseelan and T. Mitra, "Temperature aware task sequencing and voltage scaling," *International Conference on Computer Aided Design*, pp. 618–623, 2008.
[11] S. Zhang and K. S. Chatha, "System-level thermal aware design of applications with uncertain execution time," *International Conference on Computer-Aided Design*, pp. 242–249, Nov. 2008.
[12] R. Rao and S. Vrudhula, "Performance optimal processor throttling under thermal constraints," *International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*, Nov. 2007.
[13] S. Martin, K. Flautner, T. Mudge, and D. Blaauw, "Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads," *International Conference on Computer Aided Design*, pp. 721–725, Nov. 2002.
[14] W. P. Liao, L. He, and K. M. Lepak, "Temperature and supply voltage aware performance and power modeling at micro-architecture level," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. No.7, July 2005.
[15] Y. Liu, R. P. Dick, L. Shang, and H. Yang, "Accurate temperature-dependent integrated circuit leakage power estimation is easy," *Design Automation Test in Europe Conference*, 2007.
[16] M. Bao, A. Andrei, P. Eles, and Z. Peng, "Temperature-aware voltage selection for energy optimization," *Design, Automation and Test in Europe, 2008. DATE '08*, pp. 1083–1086, April 2008.
[17] Y. Nesterov and A. Nemirovskii, "Interior-point polynomial algorithms in convex programming," *Studies in Applied Mathematics*, 1994.
[18] "Application note: Powerpc 970mp thermal considerations," Jul. 2006.
[19] "Intel core 2 duo mobile processors on 65-nm process for embedded applications: Thermal design guide," Aug. 2007.
[20] "Intel core 2 duo mobile processors on 45-nm process for embedded applications: Thermal design guide," Jun. 2008.
[21] R. Jejurikar and R. Gupta, "Dynamic slack reclamation with procrastination scheduling in real-time embedded systems," *Design Automation Conference*, pp. 111 – 116, 2005.
[22] "http://ffmpeg.mplayerhq.hu/."