On-line Thermal Aware Dynamic Voltage Scaling for Energy Optimization with Frequency/Temperature Dependency Consideration

Min Bao
IDA, Linkoping University SE-58183 Linkoping, Sweden
minba@ida.liu.se

Alexandru Andrei
Ericsson, Linkoping SE-58330, Sweden
alexandru.andrei@ericsson.com

Petru Eles
IDA, Linkoping University SE-58183 Linkoping, Sweden
petel@ida.liu.se

Zebo Peng
IDA, Linkoping University SE-58183 Linkoping, Sweden
zpe@ida.liu.se

ABSTRACT

With new technologies, temperature has become a major issue to be considered at system level design. Without taking temperature aspects into consideration, no approach to energy or/and performance optimization will be sufficiently accurate and efficient. In this paper we propose an on-line temperature aware dynamic voltage and frequency scaling (DVFS) technique which is able to exploit both static and dynamic slack. The approach implies an off-line temperature aware optimization step and on-line voltage/frequency settings based on temperature sensor readings. Most importantly, the presented approach is aware of the frequency/temperature dependency, by which important additional energy savings are obtained.

Categories and Subject Descriptors

C.3 [**Special-Purpose and Application-Based Systems**]: *Microprocessor/microcomputer applications, real-time and embedded systems*; D.4.1 [**Operating Systems**]: Process Management—*scheduling*; J.6 [**Computer-Aided Engineering**]: *computer-aided design*; J.7 [**Computers in Other Systems**]: *real time*

General Terms

Algorithms, Design, Performance, Theory

Keywords

temperature dependency, energy, voltage/frequency scaling

1. INTRODUCTION

Technology scaling and ever increasing demand for performance have resulted in high power densities in current circuits, which not only results in huge energy consumption but also leads to increased chip temperature. Temperature has become a major issue to be considered at system level design. Of particular importance in this context is the development of adequate temperature modeling, analysis, and measuring tools. HotSpot [24] is an architecture and system-level temperature model and simulator, based on elaborating an equivalent circuit of thermal resistances and capacitances that correspond to the architecture blocks and to the elements of the thermal package. A similar approach is proposed in [27] where dynamic adaptation of the resolution is performed, in order to speed up the thermal analysis. Simpler, analytical temperature models, which are much less accurate, have been also proposed [23]. For on-line temperature monitoring, sensors have been used [22] together with techniques for collecting and analyzing their values with adequate accuracy [9].

Several approaches to thermal aware system-level design aiming at energy optimization or temperature control have been proposed. Static approaches are exclusively based on temperature models used at design time. For example, in [21], a simulated-annealing based thermal aware floorplanning approach has been proposed. Thermal aware task allocation and scheduling have been addressed in [26] while in [23] an approach to task scheduling under peak temperature constraints is presented. An approach to design space exploration for multiprocessor SoCs under area and thermal constraints is presented in [14].

Various techniques have been proposed in which decisions are taken based on temperature measurements on the chip at execution time. Jung [12] proposed a temperature management approach based on a Markovian decision process aiming at minimizing energy under temperature constraints. The abstract model of the managed system is constructed at design time, based on a very simple temperature model. Using this off-line generated abstract model, at run time, decisions are taken based on temperature sensor reading. No strict timing constraints are considered. The techniques proposed in [8] for dynamic OS-level workload scheduling are exclusively based on execution time temperature sensor reading. The goal is to avoid thermal hot spots and large temperature variations which have a negative impact on system reliability.

One of the preferred approaches for reducing the overall energy consumption is dynamic voltage/frequency scaling (DVFS). This technique exploits the available slack times by reducing the voltage and frequency at which the processors operate and, thus, achieves energy efficiency. There are two types of slacks: (1) static slack, which is due to the fact that, when executing at the highest (nominal) voltage level, tasks finish before their deadline even when executing their worst number of cycles (WNC); (2) dynamic slack, due to the fact that most of the time tasks execute less cycles than their WNC. Offline DVFS techniques [11], [13] can only exploit static slack while online approaches [4], [3], [25] are able to further reduce energy consumption by exploiting the variation of the workload generated by the tasks. None of the above approaches considers any implication of the chip temperature.

As mentioned earlier, high power densities on current microprocessors result in high energy consumption and increased chip 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 at voltage selection. However, very few of the proposed DVFS techniques are considering the temperature issue. In [16] an offline, static DVFS scheme is presented, aimed at reducing peak temperature. In [5] we have proposed a temperature and leakage aware offline DVFS approach for energy minimization. The approach proposed in [19] is based on a design time optimization procedure which is performed considering various start time temperatures and workloads. At run-time, frequency setting is based on actual temperatures received from sensors. The approach ignores the dependency of leakage (and, consequently, energy) on temperature and assumes (as in offline DVFS techniques) that the number of cycles executed by a given task is fixed and known at design time.

The basic idea of DVFS approaches is to achieve energy minimization by reducing the supply voltage. Since frequency depends on the voltage, every change in supply voltage has to be followed, in principle, by a frequency adjustment. In order to provide performance, the frequency is usually set to the maximum value allowed by the current supply voltage. However, temperature has an important impact on circuit delay and, implicitly, on frequency, mainly through its influence on carrier mobility and threshold voltage [7]. Thus, the frequency does not only depend on the voltage but also on the temperature. This aspect has been completely ignored even by temperature aware approaches to DVFS. In fact, when calculating the allowed frequency for a given supply voltage V_{dd}, it is implicitly assumed that this is the frequency f corresponding to a maximum temperature T_{max} at which the chip is allowed to run. While this is a safe assumption, it is far from efficient. If we are aware that the chip is running at a temperature $T < T_{max}$, the frequency could be fixed at $f' > f$ and, thus, performance is increased for the same energy consumption. Or, maybe more important, the same frequency f could be achieved with a supply voltage $V'_{dd} < V_{dd}$ and, thus, further energy is saved. As we will show in this paper, such a temperature aware dynamic frequency setting can be an important contributor to energy efficiency.

The main contribution of this paper is an online temperature aware DVFS approach which can exploit both static and dynamic slack. The approach implies an offline temperature aware optimization step and online voltage/frequency settings based on temperature sensor reading. Most importantly, the presented approach is aware of the frequency/temperature dependency by which important additional energy savings are obtained.

The paper is organized as follows: In section 2 we introduce the power, delay and application model as well as the voltage selection techniques used in the paper. Section 3 gives a motivational example. In section 4 we present our DVFS approaches. Finally, experimental results and conclusions are presented in sections 5 and 6 respectively.

2. PRELIMINARIES

2.1 Power and Delay Model

For dynamic power we use the following equation [18], [6]:

$$P_{dyn} = C_{eff} * f * V_{dd}^2 \qquad (1)$$

where C_{eff}, V_{dd}, and f denote the effective switched capacitance, supply voltage, and frequency, respectively.

The leakage power is expressed as follows [15], [18], [16]:

$$P_{leak} = I_{sr} * T^2 * e^{\frac{\alpha * V_{dd} + \beta * V_{bs} + \gamma}{T}} * V_{dd} + |V_{bs}| * I_{ju} \qquad (2)$$

where I_{sr} is the reference leakage current at reference temperature. T is the current temperature, V_{bs} is the body bias voltage, and I_{ju} is the junction leakage current. α, β and γ are curve fitting circuit technology dependent coefficients.

The frequency at a reference temperature is calculated as follows [18]:

$$f = \frac{1}{d} = \frac{((1 + k_1) * V_{dd} + K_2 * V_{bs} - v_{th1})^{\alpha}}{K_6 * Ld * V_{dd}} \qquad (3)$$

where Ld is the logic depth. K_1, K_2, K_6, and v_{th1} are technology dependent coefficients. α reflects the velocity saturation imposed by the used technology (common values $1.4 < \alpha < 2$). The scaling of frequency with temperature is given by equ.4 [15] :

$$f \propto \frac{(V_{dd} - (v_{th1} + k * (T - T_{ref})))^{\xi}}{V_{dd} * T^{\mu}} \qquad (4)$$

T is the temperature while k, ξ, and μ are empirical technology dependent constants.

2.2 Application and System Model

The functionality of the application is captured as a set of task graphs. In a task graph $G(\Pi, \Gamma)$, nodes $\tau \in \Pi$ represent computational tasks, while edges $\eta \in \Gamma$ indicate data dependencies between tasks (communication). Each task is characterized by the following parameters: the worse case (WNC), best case (BNC), and expected (ENC) number of clock cycles to be executed, a deadline, and the average switched capacitance. ENC is the arithmetic mean value of the probability density function $p(NC)$ of the task execution cycles NC, i.e., $ENC = \sum_{j=BNC}^{WNC} j * p_j(j)$

The application is mapped and scheduled on an embedded system with a voltage scalable processor which can operate at several discrete supply voltage levels. The processor has internal temperature sensors that can be accessed during execution. The application and a set of look up tables (LUT), one for each task, are stored in memory. The LUTs are generated offline and are used during execution whenever the scheduler has to adjust the processor's performance to the appropriate level, via voltage/frequency scaling. An appropriate performance level allows the tasks to meet their deadlines while maximizing the energy savings.

2.3 Temperature-Aware DVFS

In [2] we have presented a DVFS approach which given a mapped and scheduled application, calculates the appropriate voltage levels for each task, such that the total energy consumption is minimized. 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 will be used for calculating the dynamic energy consumed by the task executed at certain supply voltage levels according to equ.1. The leakage energy is calculated using equ.2. However, since leakage strongly depends on temperature, an obvious question is which temperature to use for leakage calculations. Ideally, it should be the temperature at which the chip will work when executing the application. This temperature, however, is not known, since the algorithm is just calculating the voltages at which to run the system and these voltages are influencing the energy dissipation which, again, is determining the temperature.

The algorithm in [2] requires the designer to introduce an assumed temperature which is used at energy optimization. This, of course, leads to suboptimal results, since the temperature used for energy calculation during voltage selection is different from the actual temperature at which the chip works. In order to overcome the above problem, in [5] we have proposed a temperature aware DVFS technique which is based on an iterative approach as illustrated in Fig.1. The approach starts from an initial "assumed" temperature, at which the processor is supposed 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 dynamic 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. Convergence means that the actual temperature values used at voltage selection correspond to the temperature at which the chip will function when running with the calculated voltages. As shown in [5], in most of the cases, convergence is reached in less than 5 iterations.

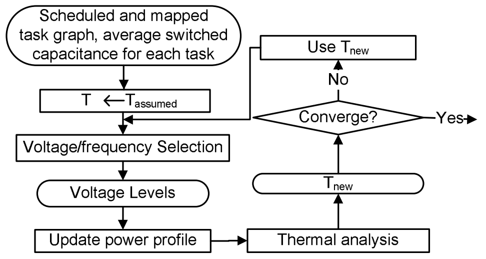

Scheduled and mapped task graph, average switched capacitance for each task
Use T_{new}
No
$T \leftarrow T_{assumed}$
Converge?
Yes
Voltage/frequency Selection
T_{new}
Voltage Levels
Update power profile
Thermal analysis

Figure 1: Temperature-aware Voltage Selection

Thermal analysis in our DVFS technique is based on the HotSpot environment [24]. Our modifications to HotSpot, in order to capture the dependence of leakage on temperature, are presented in [5].

Two limitations of the above approach are important in the context of this paper:

1. It ignores frequency/temperature dependency and, thus, produces solutions which are excessively conservative.

2. It is a static technique, assuming that tasks always execute their WNC and, thus, cannot exploit dynamic slack.

3. MOTIVATIONAL EXAMPLE

In this section, we consider an application consisting of three tasks as shown Fig.2. The WNC of τ_1, τ_2, and τ_3 is $2.85 * 10^6$, $1.0 * 10^6$, and $4.30 * 10^6$, respectively, and their average switched capacitance (in F) is $1.0 * 10^{-9}$, $0.9 * 10^{-10}$, and $1.5 * 10^{-8}$, respectively. The application has a global deadline of 0.0128s. We assume that the three tasks are executed on a processor which has 9 discrete V_{dd} levels from 1.0V to 1.8V with a step of 0.1V. The chip size is 0.007m*0.007m with a peak working temperature of $T_{max} = 125°$C.

For the above example we perform energy minimization using the temperature-aware DVFS method outlined in section 2.3. As mentioned, this DVFS approach (like all other in literature) ignores the frequency/temperature dependency and, when calculating the maximum allowed frequency for a certain supply voltage,

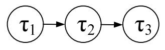


Figure 2: Motivational Example

the maximum allowed working temperature for the chip T$_{max}$= 125°C is considered. In Table 1 we show the actual voltages and frequencies for each task, as calculated by the DVFS algorithm, and the consumed energy. We also show the peak temperature for each task, obtained with dynamic thermal analysis. As can be observed, this peak temperature is far below the T$_{max}$ of the chip.

Table 1: DVFS without Frequency/Temperature Dependency

Task	Peak Temp(°C)	Voltage(V)	Freq(MHz)	Energy(J)	Total
τ_1	74.6	1.8	717.8	0.063	
τ_2	73.3	1.7	658.8	0.017	
τ_3	74.7	1.6	600.1	0.228	0.308

From equ. 3 and 4 it is obvious that by taking into consideration the actual temperature at frequency calculation there is a large margin for reducing the supply voltage without compromising on performance. We have performed a DVFS based energy optimization, similar to the one above, but with the difference that frequencies corresponding to the different voltage settings are calculated taking into consideration the peak temperature at which the actual task runs. Table 2 shows the results. We can see that a substantial energy reduction of 33% has been obtained.

Table 2: DVFS with Frequency/Temperature Dependency

Task	Peak Temp(°C)	Voltage(V)	Freq(MHz)	Energy(J)	Total(J)
τ_1	61.1	1.8	836.7	0.051	
τ_2	59.9	1.7	765.1	0.013	
τ_3	61.1	1.3	483.9	0.142	0.206

The DVFS approach used above is an off-line, static one which assumes that tasks execute their WNC and, thus, can only exploit the static slack. However, in reality, there are huge variations in the number of cycles executed by a task, from one activation to the other, which leads to an important amount of dynamic slack. Imagine an activation scenario for which each of the three tasks in Fig. 2 execute a number of cycles equal to 60% of their WNC. If we use the offline DVFS approach above and run at the voltages calculated as in Table 2 the total energy consumption would be 0.122J. However, much more can be done by also exploiting the dynamic slack. This implies that, at run-time, whenever a task terminates, the voltage level and the frequency for the next task is calculated by taking into consideration the current time and the current chip temperature. Table 3 shows the voltage and frequency levels determined in this way as well as the corresponding energy consumption. The total energy consumed is 0.106J, which means a reduction of 13.1% compared to the off-line DVFS approach.

The examples presented in this section demonstrate that (1) considering the frequency/temperature dependency at DVFS can lead to substantial energy savings and (2) an on-line temperature aware approach is needed in order to make use of the dynamic slack created due to variable number of clock cycles executed at different activations.

4. DVFS WITH FREQUENCY/TEMPERATURE DEPENDENCY

4.1 Static Approach

The static approach is based on the technique outlined in section 2.3 and described in detail in [5]. As can be observed in Fig. 1, the successive iterations are leading, after convergence, to a temperature profile which corresponds to the one at which the chip will work. This temperature profile is used for energy calculation by the voltage/frequency selection algorithm.

For each task τ_i the voltage/frequency selection algorithm calculates a certain supply voltage V_{dd_i} such that energy consumption is minimized and deadlines are satisfied. When calculating the frequency setting for τ_i, as opposed to [5] and all other previous approaches, we now consider the thermal profile of the task and determine the maximum temperature T_{peak_i} at which that task runs. At voltage/frequency selection, the frequency is calculated

Table 3: Dynamic DVFS

Task	Peak Temp(°C)	Voltage(V)	Freq(MHz)	Energy(J)	Total(J)
τ_1	50.5	1.5	625.2	0.018	
τ_2	50.4	1.5	625.2	0.005	
τ_3	51.4	1.3	481.2	0.083	0.106

based on equ. 3 and 4 (instead of being fixed, in a conservative way, considering the worst case temperature T_{max} for which the chip is designed).

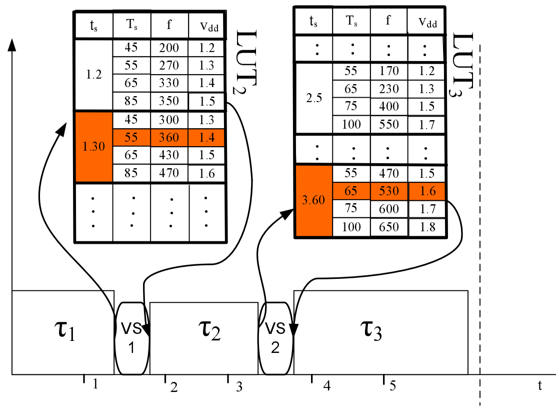
4.2 Dynamic Approach

The above approach determines start times for tasks and their voltage/frequency levels assuming that they execute their WNC. By this, only static slack is considered for energy minimization [1].

In order to exploit the dynamic slack, at the termination of each task and before starting the next one, voltage and frequency settings have to be determined based on the values of the current time and temperature. In principle, calculating the appropriate voltage/frequency settings implies the execution of the temperature aware DVFS algorithm from section 4.1. Running this algorithm on-line, after each task execution, implies a huge time and energy overhead which can be even higher than the execution time and energy consumption of the actual application.

To overcome the above problem, we have divided our dynamic DVFS approach into two phases. In the first phase, performed off-line, voltage/frequency settings for all tasks are pre-computed, based on possible start times of the tasks and the possible temperatures at that start time. The resulting voltage/frequency settings are stored in look-up tables (LUTs), one for each task. In Fig. 3 we show two such tables. They contain voltage and frequency settings for combinations of possible start time t_s and start temperature T_s of a task. For example, the line with start time 1.3 and start temperature 55 stores the voltage and frequency setting for the situation when τ_2 starts in the time interval (1.2s,1.3s] and the start temperature is in the interval (45°C,55°C]. In section 4.2.1 we will present the generation procedure of the LUTs.

The second phase is performed on-line and is illustrated in Fig. 3. Each time a task terminates and a new voltage/frequency level has to be fixed for the next one, the on-line scheme looks up the appropriate setting from the LUT, depending on the actual time and temperature reading. If there is no exact entry in the LUT corresponding to the actual time/temperature, the entry corresponding to the immediately higher time/temperature is selected. For example, in Fig.3, τ_1 finishes at time 1.25s with a temperature 49°C. To determine the appropriate voltage and frequency for τ_2, LUT$_2$ is accessed based on these time and temperature values. There is no exact entry for 1.25s and 49°C, so the entry corresponding to start time 1.3s and start temperature 55°C is chosen. This on-line phase indicated with VS at Fig. 3 is of very low, constant time complexity O(1) and, thus, very efficient.



t_s	T_s	f	V_{dd}
1.2	45	200	1.2
	55	270	1.3
	65	330	1.4
	85	350	1.5
1.30	45	300	1.3
	55	360	1.4
	65	430	1.5
	85	470	1.6
⋮	⋮	⋮	⋮

t_s	T_s	f	V_{dd}
:	:	:	:
2.5	55	170	1.2
	65	230	1.3
	75	400	1.5
	100	550	1.7
:	:	:	:
3.60	55	470	1.5
	65	530	1.6
	75	600	1.7
	100	650	1.8

Figure 3: On-line Phase

4.2.1 *LUT Generation*

Considering an application and a system as described in section 2.2, the goal is to generate a LUT for each task τ_i, such that the

[1]It should be mentioned that, as opposed to the dynamic one, the static approach can be used even in the case that no temperature sensors are available on the chip.

energy consumption during execution is minimized. The task execution order is fixed according to a scheduling policy (e.g. EDF). According to this order, task τ_i has to be executed after τ_{i-1} and before τ_{i+1}. It is important to notice that the voltage levels and frequencies are calculated so that the energy consumption is optimal in the case that the tasks execute their expected number of cycles ENC (which, in reality, happens with a much higher probability than e.g. the WNC). Nevertheless, voltages and frequencies are fixed such that, even in the worst case (tasks execute WNC), deadlines are satisfied.

The LUT generation algorithm is presented in Fig.4. The outermost loop iterates over the set of tasks and successively constructs the table LUT$_i$ for each task τ_i. The next loop generates the entries of LUT$_i$ corresponding to the various start temperatures T_{s_i} of τ_i. Finally, the innermost loop iterates, for each possible start temperature, over all considered start times t_{s_i} of task τ_i. The algorithm starts by computing the earliest and latest possible start times for each task. The earliest start time EST_i is calculated based on the situation that all tasks execute with their best case number of cycles BNC at the highest voltage setting and lowest temperature (the ambient temperature). The latest start time LST_i is calculated as the latest start time of τ_i that still allows to satisfy the deadlines for all tasks τ_j, $j \geq i$, when executed with the worst case number of cycles WNC at the highest voltage and the maximum temperature T_{max} allowed for the chip.

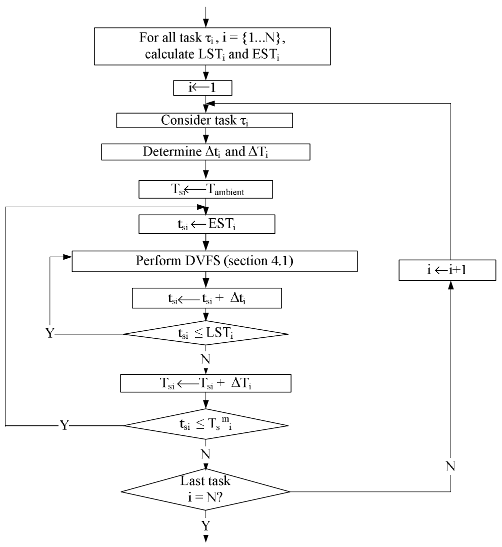

For all task τ_i, i = {1...N}, calculate LST$_i$ and EST$_i$
i⟵1
Consider task τ_i
Determine Δt_i and ΔT_i
T_{si}⟵$T_{ambient}$
t_{si}⟵EST$_i$
Perform DVFS (section 4.1)
t_{si}⟵t_{si} + Δt_i
Y
$t_{si} \leq$ LST$_i$
N
T_{si}⟵T_{si} + ΔT_i
Y
$t_{si} \leq T_s{}^m{}_i$
N
Last task i = N?
Y
N
i⟵i+1

Figure 4: LUT generation

Considering the intended granularity of the LUT, the time and temperature quanta $\triangle t_i$ and $\triangle T_i$ are determined. Thus, for task τ_i, the number of time entries (the number of different start times considered) will be $(LST_i - EST_i)/\triangle t_i$, while, for each possible start time, the number of temperature entries is $(T_s{}^m{}_i - T_{ambient})/\triangle T_i$, where $T_s{}^m{}_i$ is the maximum possible temperature at the start time of τ_i. In sections 4.2.2 and 4.2.3 we will further elaborate on the granularity and size of the LUTs.

When calculating the actual LUT entries for a task τ_i, the calculation of the voltage and frequency setting is performed by running the DVFS algorithm outlined in section 4.1, for all tasks τ_j, $j \geq i$, considering t_{s_i} and T_{s_i} as start time and starting temperature, respectively, for τ_i.

4.2.2 Temperature Bounds and Granularity

As discussed before, the number of entries generated in LUT$_i$ along the temperature dimension is $(T_s{}^m{}_i - T_{ambient})/\triangle T_i$. The basic idea is that the lowest possible temperature is the temperature of the ambient, while $T_s{}^m{}_i$ is the highest possible temperature, in the worst case, at the start time of task τ_i. But what is the value of $T_s{}^m{}_i$? One solution is to consider for $T_s{}^m{}_i$ the maximum temperature T_{max} at which the chip is allowed to work. While this assumption is safe, it leads to unnecessarily large tables since, during the execution of most of the tasks, the chip will never reach temperatures close to T_{max}. In order to avoid unnecessarily large tables, we need a safe but tighter upper bound on the temperature $T_s{}^m{}_i$. In order to achieve this goal, our LUT generation algorithm in Fig.4 is executed several times in successive iterations before the final LUT tables are obtained.

We start by considering that for the first task the maximal starting temperature is the ambient temperature ($T_s{}^m{}_1 = T_{ambient}$). The two inner loops in Fig. 4 will generate LUT$_1$. As part of the DVFS procedure executed during generation of LUT$_1$ we obtain the possible temperature profiles of τ_1 and, thus, also the peak temperature T_{peak_1} reached during execution of this task. The worst case starting temperature of task τ_2 is $T_s{}^m{}_2 = T_{peak_1}$. Considering this value for $T_s{}^m{}_2$, table LUT$_2$ is generated and the procedure is continued for all tasks τ_i. After the algorithm in Fig. 4 has been executed once, we have all LUT tables, based on the assumption that the maximal possible temperature at the start of τ_1 is equal to $T_{ambient}$. This, however, is not the case, since the application is executed periodically and τ_1 is started again after the last task τ_N. Thus, in fact, the maximal staring temperature of τ_1 is, in the worst case, equal to the worst case peak temperature of τ_N. Therefore, we repeat the LUT generation algorithm, this time considering that $T_s{}^m{}_1 = T_{peak_N}$. This will lead to a higher T_{peak_1} than in the previous iteration and, thus, a new larger $T_s{}^m{}_2 = T_{peak_1}$. Thus, new lines will be generated in the LUTs. The procedure is continued iteratively, until, for a certain task, the peak temperature over two successive iterations does not change, which means that no new entries into the LUT tables will be generated. Our experiments have shown that convergence is reached after not more then 3 iterations. This procedure also allows to detect if there exists a possibility for the design to reach, in the worst case, a thermal runaway situation (in which case the iterations do not converge) or if the maximum allowed temperature can be violated (there is convergence but there are peak temperatures which are beyond T_{max}).

The above technique leads to a tightening of the range of temperatures in the LUT. There are two more questions to be answered regarding the number of temperature entries (1) What should be the granularity of the temperature investigation and (2) how to reduce the number of entries if only a limited amount of memory is available?

It is obvious that a finer granularity and larger number of entries will, potentially, produce better energy savings at the cost, however, of increased memory consumption. With regard to the granularity $\triangle T_i$, our experiments have shown that values around 15°C are optimal, in the sense that finer granularities will only marginally improve energy efficiency. If, due to memory limitations, we only can afford a certain number of temperature entries NT_i to be stored for a task τ_i, we have to decide which lines of LUT$_i$ to preserve and which to eliminate. One straightforward approach would be to maintain an even distribution of the selected NT_i lines over the range $[T_{ambient}, T_s{}^m{}_i]$. However, start temperatures of tasks, during execution, do not spread evenly over this range. Thus, it is more efficient to have the NT_i lines more dense around the temperature values that are more likely to happen, and sparse towards the extremes. This means that less pessimistic voltage/frequency settings will be used for the most likely cases, while cases that are much less likely to happen are handled in a more pessimistic way. Thus, after the LUT tables have been generated, in order to select the appropriate NT_i lines along the temperature dimension for each task τ_i, we run a temperature analysis session in which all tasks are executed for their expected number of cycles ENC. From this analysis, we can observe which is the most likely starting temperature for each task and we select the NT_i lines among those close to this most likely temperature.

4.2.3 LUT Granularity Along the Time Dimension

A straightforward approach would be to allocate the same number of entries, along the time dimension, to each task (Nt_i is the same for all tasks τ_i, $i = 1..N$). However, the start time interval sizes $LST_i - EST_i$ can differ very much between tasks, which should be taken into consideration when deciding on the number of time entries. Therefore, given a total number of entries along the time dimension NL_t, we determine the number of time entries in each LUT$_i$, as follows [2]:

$$Nt_i = NL_t * \frac{(LST_i - EST_i)}{\sum_{i=1}^{N}(LST_i - EST_i)} \qquad (5)$$

4.2.4 Accounting for Analysis Accuracy and Ambient

The solutions produced by our techniques presented in section 4.1 and 4.2 are safe. By this we mean that:

[2]Let us mention that while the start time intervals are very different from task to task, this is much less the case with temperature interval. Therefore, the number of entries along the temperature dimension (NT_i, see section 4.2.2 has been kept identical for all tasks in our experiments.

1. It is guaranteed that deadlines are satisfied;

2. If, at run time, a certain frequency setting is selected for a task τ_i, it is guaranteed that the temperature during execution of τ_i will not exceed the limit allowed for the chip to run at the selected frequency.

There are two aspects which have to be discussed with respect to the second of the two statements above. First is the issue of ambient temperature. If a task τ_i is starting its execution at a certain temperature T, the temperature profile during task execution depends on the actual ambient temperature. Thus, a safe frequency selection has to also take into consideration the current ambient temperature. Two possible solutions can be considered:

1. Generate the voltage/frequency settings considering the highest ambient temperature under which the system is supposed to function. This is a safe but pessimistic solution with, potentially, smaller energy savings.

2. Generate alternative voltage/frequency settings for a set of ambient temperatures in the range assumed for the system to function. During run time, using sensors for the ambient temperature, the system will switch to those tables corresponding to that ambient temperature that is immediately higher than the actual measured one. This solution requires additional memory for storing a larger amount of tables but could lead to better energy efficiency.

The second aspect to be considered is the accuracy of the temperature analysis. The fact that a certain frequency setting is safe, with regard to the peak temperature reached during execution of a task, is based on the temperature analysis performed as part of the DVFS procedure. Thus, the results can be safe only to the extent to which this analysis provides safe temperatures. Of course, system level thermal analysis tools are not provably accurate. Nevertheless, relative precisions are reported for the various analysis tools and we are using this information in order to account for the inaccuracy of the thermal analysis. More precisely, given a certain relative precision of the temperature analysis tool that we use, we account for this precision in a conservative way when determining the peak temperatures used for frequency calculation.

In section 5, we will evaluate the impact of both ambient temperature and potential analysis inaccuracy on the energy optimization results.

5. EXPERIMENTAL RESULTS

Our experiments have been performed on both generated applications and a real-life example.

We have randomly generated applications consisting of 2 to 50 tasks. The WNC of the tasks are in the range [10^6,10^7]. The applications are executed on a processor which can run at 9 different supply voltage levels in the range [1.0, 1.8]. The temperature model related coefficients are the same as in [15], while the power model related coefficients are as in [18]. Similar to [15] and [20], in equ.4 we use the coefficients $\mu = 1.19$, $\xi = 1.2$, and k = -1.0V/°C. If not mentioned differently we assume an ambient temperature of 40°C.

It is important to mention that in all our experiments, we have accounted for the time and energy overhead produced by the on-line component of our dynamic approach. Similarly, we have also taken into consideration the energy overhead due to the memories. This overhead has been calculated based on the energy values given in [10] and [17].

The first set of experiments is aimed at evaluating the efficiency of taking into consideration the frequency/temperature dependency. We first compare the static DVFS approach in [5], which ignores the frequency/temperature dependency, to our static approach proposed in section 4.1. Considering the average over all 25 applications the energy consumption obtained by considering frequency/temperature dependency is 22% smaller than when ignoring this dependency.

For the dynamic approach, described in section 4.2, we have run the same set of experiments, first ignoring frequency/temperature dependency and than considering it. The energy consumption has been reduced, on average, by 17% in the latter case.

The next set of experiments compares the energy consumption with the static DVFS approach in section 4.1 and the dynamic one in section 4.2 (both considering frequency/temperature dependency). As the ratio BCN/WCN has a strong influence on the potential efficiency of a dynamic approach, we run the experiments considering three different ratios: 20%, 50%, and 70%. Also, we assume that the workload distribution of each task conforms to a normal distribution $N(ENC, \sigma^2)$, where ENC is the mean value, and σ is the standard deviation. For our energy evaluations we have generated actual numbers of executed clock cycles for each task considering standard deviations of $(WNC - BNC)/3$, $(WNC-BNC)/5$, $(WNC-BNC)/10$, and $(WNC-BNC)/100$. Fig.5 shows the energy savings with the dynamic approach relative to the static one. As can be observed, the efficiency of the dynamic approach, compared to the static one, increases as the ratio between BNC and WNC becomes smaller. Remember that our DVFS algorithm is targeted torwards optimizing the energy consumption for the case that tasks execute the expected nunmber of cycles ENC. Therefore, energy savings are larger, compared to the static approach, when the standard deviation σ is smaller (more of the actual executed number of clock cycles are clustering around the ENC).

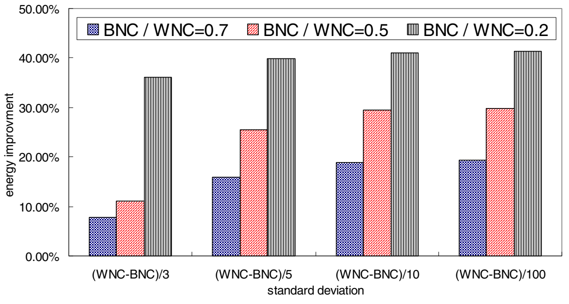


Figure 5: Dynamic vs Static Approach

The third set of experiments is aimed at exploring the impact of the LUT sizes. In particular, for this paper, we are interested in the impact of entries along the temperature dimension. The number of lines along the time dimension has been kept constant for these experiments and is distributed according to the discussion in section 4.2.3. First we run, for all applications, our dynamic DVFS approach considering a granularity $\triangle T = 10^\circ$. We evaluate the average energy reduction with the obtained LUTs, compared to the static approach. Then we impose a certain limitation on the number of lines along the temperature dimension and we construct the corresponding LUTs as discussed in section 4.2.2. We again evaluate the energy consumption considering these reduced LUTs. The diagram in Fig. 6 shows the average results for different number of lines and two different standard deviations of the actual number of clock cycles executed by tasks. Having one single temperature entry will produce energy reductions compared to the static case which are 37% smaller (for $\sigma = (WNC - BMC)/3$) than with an unreduced LUT. However, with 2 entries the results are already very close to those obtained with an unreduced LUT and with 3 entries they are, in practice, identical. This is good news, since it shows that significant energy savings can be obtained with relatively small memory overhead. It should be mentioned that all other experiments presented in this section have been performed with 2 entries along the temperature dimension.

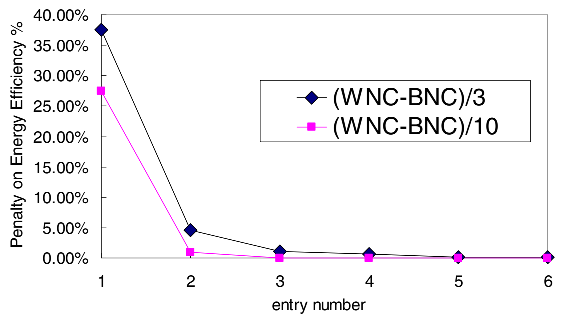


Figure 6: Impact of Temperature Line Number

Our final experiments have been performed in order to explore the impact of ambient temperature and temperature analysis accuracy. For all experiments above, we have assumed that $T_{ambient}$ is 40°C and is known at design time. In order to evaluate the impact of the ambient, we considered all the generated applications and constructed LUTs for values of $T_{ambient}$ in the range [-10°C, 40°C]. For each (application, LUTs) pair corresponding to a certain $T_{ambient}$ we evaluated the energy consumption considering that the $T_{ambient}$ is identical with the one assumed at LUT generation. Then we run the simulations for the same (application, LUTs) pair, but considering that $T_{ambient}$ deviates with 10°, 20°,...,50° from the value assumed at design time. The results are shown in Fig.7. We can see that if $T_{ambient}$ is different by, for example, 20° from the one assumed at design time, the energy consumption increases by only 7% on average. This shows that, if the predicted range of ambient temperature is, for example, 40°, generating two sets of LUTs (granularity of 20°) will lead to energy losses, on average, less than 7%.

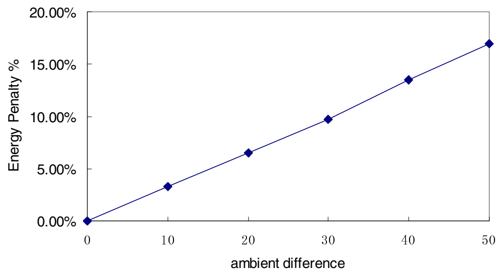


Figure 7: Impact of the Ambient Temperature

All the presented experiments have been performed considering that the temperature modeling and analysis is accurate. We have repeated the experiments considering a relative accuracy of 85%. When calculating frequency settings we accounted, in a conservative way, for this degree of accuracy. Our experiments have shown that the energy degradation due to the 85% relative accuracy is less than 3%.

Finally, we apply our static and dynamic approach described in section 4 to a real life case, namely an MPEG2 decoder which consists of 34 tasks and is described in more detail in [1]. The energy consumption using the static approach is reduced by 22% when considering frequency/tempeature dependency. For the dynamic approach the reduction is 19%. When considering frequency/temperature dependency, the dynanmic approach gives an energy reduction of 39% compared to the static one.

6. CONCLUSION

We have proposed an on-line temperature aware dynamic voltage and frequency scaling (DVFS) technique which is able to exploit both static and dynamic slack and takes into consideration the frequency/temperature dependency. The approach consists of two parts: an off-line temperature aware optimization step and on-line voltage/frequency setting based on temperature sensor readings. Experiments show that significant energy improvements can be achieved using the proposed techniques.

7. REFERENCES

[1] http://ffmpeg.mplayerhq.hu/.

[2] A. Andrei, P. Eles, Z. Peng, M. Schmitz, and . . B. M. Al-Hashimi. , energy optimization of multiprocessor systems on chip by voltage selection. *IEEE Transactions on Very Large Scale Integration Systems*, 15((3)):pp.262–275.

[3] A. Andrei, M. Schmitz, P. Eles, Z. Peng, and B. Al-Hashimi. Quasi-static voltage scaling for energy minimization with time constraints. *Design Automation and Test (DATE)*, April 2005.

[4] H. Aydin, R. Melhem, D. Moss, and P. Alvarez. Dynamic and aggressive scheduling techniques for power-aware real-time systems. *22nd IEEE Real-Time Systems Symposium (RTSS'01)*, pages pp. 95– 105, Dec. 2001.

[5] M. Bao, A. Andrei, P. Eles, and Z. Peng. Temperature-aware voltage selection for energy optimization. *Design Automation and Test (DATE)*, April 2008.

[6] A. P. Chandrakasan and R. W. Brodersen. *Low Power Digital CMOS Design.* Norwell, MA: Kluwer, 1995.

[7] R. Cobbold. Temperature effects on mos transistors. *Electronic Letters*, 2:pp. 190ÍC192, 1966.

[8] A. Coskun, T. Rosing, and K. Whisnant. Temperature aware task scheduling in mpsocs. *Design, Automation & Test in Europe Conference & Exhibition, 2007. DATE '07*, (No.7):pp. 1–6, April 2007.

[9] K. Gross, K. Whisnant, and A. Urmanov. Electronic prognostics through continuous system telemetry. *In 60th Meeting of the Society for Machine Failure Prevention Technology (MFPT)*, (1):pp. 53ÍC62, Apr. 2006.

[10] S. Hsu and A. A. et al. A 4.5-ghz 130-nm 32-kb l0 cache with a leakage-tolerant self reverse-bias bitline scheme. *IEEE JOURNAL of Solid-State Circuits*, May 2003.

[11] T. Ishihara and H. Yasuura. Voltage scheduling problem for dynamically variable voltage processors. *Low Power Electronics and Design, 1998. Proceedings. 1998 International Symposium on*, pages pp. 197– 202, Aug. 1998.

[12] H. Jung, P. Rong, and M. Pedram. Stochastic modeling of a thermally-managed multi-core system. *Design Automation Conference, 2008*, pages pp. 728–733, June 2008.

[13] W. Kwon and T. Kim. Optimal voltage allocation techniques for dynamically variable voltage processors. *ACM TECS*, 4(1):pp. 211–230, 2005.

[14] Y. Li, B. C. Lee, D. Brooks, Z. Hu, and K. Skadron. Cmp design space exploration subject to physical constraints. *HPCA06*, pages pp. 15–26, 2006.

[15] W. P. Liao, L. He, and K. M. Lepak. Temperature and supply voltage aware performance and power modeling at micro-architecture level. *IEEE TonCAD*, 24(No.7):pp. 1042ÍC1053, July 2005.

[16] Y. Liu, H. Yang, R. Dick, H. Wang, and L. Shang. Thermal vs energy optimization for dvfs-enabled processors in embedded systems. *Symp. on Quality Electronic Design (ISQED07)*, (International Symposium on Quality Electronic Design, 2007. ISQED '07. 8th):pp. 204–209, Mar. 2007.

[17] A. Macii, E. Macii, and M. Poncino. Improving the efficiency of memory partitioning by address clustering. *Design, Automation and Test in Europe Conference and Exhibition,*, 2003.

[18] S. Martin, K. Flautner, T. Mudge, and D. Blaauw. Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors. *ICCAD, .*, pages pp. 721–725, 2002.

[19] S. Murali, A. Mutapcic, and D. A. et al. Temperature control of high-performance multi-core platforms using convex optimization. *JOURNAL of VLSI Signal Processing Systems*, (Design, Automation and Test in Europe, 2008. DATE '08):pp. 110–115, Mar. 2008.

[20] B. Razavi. *Design of Analog CMOS Integrated Circuits.* McGraw-Hill Science Engineering, Erewhon, NC, August 2000.

[21] K. Sankaranarayanan, S. Velusamy, and K. S. M.R. Stan. A casefor thermal-aware floorplanning at the microarchitectural level. *the JOURNAL of Instruction-Level Parallelism*, 7:pp. 1–16, Oct 2005.

[22] M. Sasaki, M. Ikeda, and K. Asada. -1/+0.8°c error, accurate temperature sensor using 90nm 1v cmos for on-line thermal monitoring of vlsi circuits. *Microelectronic Test Structures, 2006 IEEE International Conference*, pages pp.9–12, March 2006.

[23] S. Wang and R. Bettatin. Delay analysis in temp.-constrained hard real-time systems with general task arrivals. *RTSS06*, pages pp. 323–334, 2006.

[24] 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 on VLSI Systems*, 14(5):pp. 501–513, May 2006.

[25] C. Xian, Y.-H. Lu, and Z. Li. Dynamic voltage scaling for multitasking real-time systems with uncertain execution time. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(8):pp. 1467–1478, Aug. 2008.

[26] Y. Xie and W.-L. Hung. Temperature-aware task allocation and scheduling for embedded multiprocessor systems-on-chip (mpsoc) design. *JOURNAL of VLSI Signal Processing Systems*, 45(3):pp. 177–189, Dec. 2006.

[27] Y. Yang and Z. G. et al. Isac: Integrated space and time adaptive chip-package thermal analysis. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Jan. 2007.