



CISTER

Research Center in
Real-Time & Embedded
Computing Systems

Technical Report

Applying Idealised DVFS Algorithms to Thermally Constrained DPM

Muhammad Ali Awan

Stefan M. Petters

CISTER-TR-130601

Version:

Date: 06-05-2013

Applying Idealised DVFS Algorithms to Thermally Constrained DPM

Muhammad Ali Awan, Stefan M. Petters

CISTER Research Unit

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8340509

E-mail:

<http://www.cister.isep.ipp.pt>

Abstract

Modern real-time embedded systems have increasingly penetrated our daily life and are also often constrained in terms of temperature and energy. In this paper, a thesis is defended that from real-time systems perspective, thermallyconstrained dynamic power management approaches behave very similar to idealised dynamic voltage and frequency scaling. Hence, existing dynamic voltage and frequency scaling solutions proposed for periodic/sporadic task models can be applied to thermally constrained dynamic power management systems with moderate effort. A detailed discussion is presented that shows the similarities along with the distinctive elements between two approaches. Within the case study, the porting of a dynamic voltage and frequency scaling algorithm of the literature to thermally constrained dynamic power management system is demonstrated. The proof of concept and effectiveness of the proposed approach is shown with the help of extensive simulations performed by varying system-specific parameters across different dimensions.

Applying Idealised DVFS Algorithms to Thermally Constrained DPM

Muhammad Ali Awan Stefan M. Petters
CISTER/INESC-TEC, ISEP, Polytechnic Institute of Porto, Portugal
muaan,smp@isep.ipp.pt

Abstract—Modern real-time embedded systems have increasingly penetrated our daily life and are also often constrained in terms of temperature and energy. In this paper, a thesis is defended that from real-time systems perspective, thermally constrained dynamic power management approaches behave very similar to idealised dynamic voltage and frequency scaling. Hence, existing dynamic voltage and frequency scaling solutions proposed for periodic/sporadic task models can be applied to thermally constrained dynamic power management systems with moderate effort. A detailed discussion is presented that shows the similarities along with the distinctive elements between two approaches. Within the case study, the porting of a dynamic voltage and frequency scaling algorithm of the literature to thermally constrained dynamic power management system is demonstrated. The proof of concept and effectiveness of the proposed approach is shown with the help of extensive simulations performed by varying system-specific parameters across different dimensions.

I. INTRODUCTION

The increase in power density of modern processors demands efficient thermal management solutions to keep the temperature within given limits to avoid physical damage and also to increase the reliability of the chip. Thermal management can be done at design time through sophisticated packaging and heat dissipation techniques, and at run time through dynamic thermal management (DTM). However, packaging and active heat dissipation solutions are very expensive [1]. It has been predicted that the packaging solutions will become more challenging in the near future due to an increase in peak power and the high power density in emerging system-in-packages. This trend motivates to explore DTM techniques for a wide variety of systems.

Energy consumption is another important concern in the design process with the objective to prolong the battery life of embedded systems. The increased energy demand, due to further integration can lead to an increase in the size of embedded system which is not desirable in many cases such as mobile phones. Furthermore, a longer lasting battery is a market differentiator. Energy efficiency has the objective to reduce the cumulative power dissipation, while DTM techniques aim to keep the peak temperatures of the processor below the critical limit. A large amount of work exists dealing with both issues in a non real-time setting summarised by Kong et al. [2]. However, the problem is exacerbated with additional timing constraints of real-time (RT) systems, which are required to be met on top of functional aspects for the overall system to be considered correct.

The commonly used DTM approaches in RT systems to handle the thermal constraint along with energy and temporal restrictions are speed scheduling and thermally constrained dynamic power management (TCDDPM). *Speed Scheduling*: The frequency of the processor is reduced to decrease the temperature and the dynamic power consumption of the system. *TCDDPM*: The processor executes the workload at full speed and switches off when the peak temperature is reached to cool down the system. In the context of this research effort only TCDDPM approach is considered.

The state-of-the-art has mostly focused on the objective to reduce the peak temperature of the system under performance constraints [3], [4], [5], [6], [7]. For instance, Chaturvedi et al. [6] developed a leakage-aware scheduling algorithm called *m*-oscillating for frame-based (same period tasks) periodic hard RT systems to minimise the peak temperature. Given 2-speed schedule, their *m*-oscillating algorithm divides the high-speed level and low-speed interval into *m* sections, and run these sections alternatively. The maximum temperature decreases with an increase in *m*. Similarly, the temporal aspects (schedulability) of the hard RT systems are explored by Quan et al. [8].

Another area of RT research in this domain is the energy minimisation under thermal constraint. For example, Huang and Quan [9] extended the *m*-oscillating algorithm [6] to reduce the energy consumption of the frame-based RT system. They derived the energy function in the form of *m* and obtained its optimal value with an exhaustive search under the given temperature constraint.

Recently, it has been shown that leakage power consumption is temperature dependent and increases rapidly with a rise in temperature [10]. Yuan et al. [11] proposed the online temperature-aware leakage minimisation technique TALK for frame-based RT systems. The basic idea is to execute workload when the processor is cool and postpone it at high temperature. A pattern based approach [12] reduces the energy consumption of the frame-based RT systems with a temperature dependent leakage-power consumption. This approach divides the given frame (Time Horizon) into several equally-sized time-segments. The execution of the task is performed in the beginning of each time-segment and followed by a cooling phase using a low power sleep state. The required execution of the system and the idle time is equally divided among the time-segments. They developed a procedure to determine the optimal pattern that minimise the energy consumption.

The state-of-the-art though addresses the various aspects of RT systems under thermal constraints but makes some of the following assumptions: i) frame based (same period tasks) RT system, ii) leakage power consumption is independent of temperature, iii) ignore energy consumption. This paper presents the detailed study on the equivalence of idealised DVFS with TCDPM and shows conventional idealised DVFS algorithms can be applied with minimal modifications to TCDPM to reduce the energy consumption of the system. A realistic power model is considered with temperature dependent leakage current. The equivalence shown in this work, relaxes the restriction of identical period tasks (frame-based systems) and allows generic workload model such as sporadic tasks model without any additional complexity in the analysis.

The rest of the paper is organised as follows: **Section II** describes the system model that includes workload model, power model and thermal model. **Section III** and **Section IV** present the preliminaries needed for the discussion on the equivalence of DVFS and TCDPM given in **Section V**. The case study shown in **Section VI** ports the existing DVFS algorithms to TCDPM. The implementation concerns are discussed in **Section VII**. The evaluation of the given approach is performed in **Section VIII**. The concluding remarks along with the future directions are presented in **Section IX**.

II. SYSTEM MODEL

This section presents the workload/task-set model and the characteristics of the underlying hardware. The power and the thermal model used in this paper are adopted from the work of Yang et al. [12].

A. Workload Model

This work assumes a hard RT system, where a system cannot afford to miss any deadline. The workload consists of a task-set τ of ℓ independent sporadic tasks i.e. $\tau \stackrel{def}{=} \{\tau_1, \tau_2, \dots, \tau_\ell\}$. A task τ_i is characterised by a 3-tuple $\langle C_i, D_i, P_i \rangle$, where C_i is the worst-case execution time (WCET), D_i is the relative deadline and P_i is the minimum inter-arrival time of the tasks. This work can be extended for $D_i < P_i$, however, for the ease of presentation it is assumed $D_i = P_i$. The optimal uniprocessor Earliest-Deadline-First (EDF) dynamic priority algorithm is used to schedule a task-set τ . A Task τ_i has an individual utilisation of $U_i \stackrel{def}{=} \frac{C_i}{P_i}$ and the overall system utilisation is defined as $U \stackrel{def}{=} \sum_{i=1}^{\ell} U_i$. Each task τ_i releases potentially an sequence of infinite jobs $j_{i,m}$. A job $j_{i,m}$ of a task τ_i may execute for less than its C_i . The actual execution time of $j_{i,m}$ is denoted as $c_{i,m}$.

B. Power Model

The leakage-current is considered to be temperature dependent. The average leakage current $\bar{I}(T, V_{dd})$ at temperature T and Supply voltage V_{dd} is modelled by Laio et al. [10] as given in **Equation 1**,

$$\bar{I}(T, V_{dd}) = \bar{I}(T_0, V_0) \left(AT^2 e^{\left(\frac{\alpha V_{dd} + \beta}{T}\right)} + B e^{(\gamma V_{dd} + \delta)} \right) \quad (1)$$

where $\alpha, \beta, \gamma, \delta, A$ and B are empirical constants. $\bar{I}(T_0, V_0)$ is a reference leakage current on temperature T_0 with a reference supply voltage of V_0 . The unit of temperature is in Kelvin (K). It is based on the curve fitting of the power consumption of the different circuit types at different temperatures with SPICE simulations. Yang et al. [12] found a good approximation of such modelling in a quadratic form as shown in **Equation 2**,

$$\bar{I}(T, V_{dd}) = \hat{A}T^2 + \hat{B} \quad (2)$$

$$\hat{A} = \frac{\bar{I}(T_H, V_{dd}) - \bar{I}(T_L, V_{dd})}{T_H^2 - T_L^2} \quad (3)$$

$$\hat{B} = \bar{I}(T_L, V_{dd}) - \hat{A}T_L^2 \quad (4)$$

where \hat{A} and \hat{B} are constants, while T_H and T_L define the operating temperature range of the chip. They showed difference of this approximation is negligible when compared to average leakage current modelled by Laio et al. [10] (**Equation 1**).

The processor assumed in this work has two modes: *active* and *sleep state*. The execution of tasks is performed in the active mode and P_a denotes its power consumption. It has two components: a) dynamic power consumption P_{dyn} and b) static or leakage power consumption P_{lkg} . The dynamic power consumption of the processor is considered constant in active mode, while the static power consumption is modelled as $P_{lkg} = AT^2 + B$, where A and B are $N_{gate}\hat{A}V_{dd}$ and $N_{gate}\hat{B}V_{dd}$ respectively. N_{gate} is a constant that depends on the circuit characteristics (for more details refer to [12], [10]). The system can transition to a sleep state for two different purposes: 1) to cool down the processor and 2) to reduce the energy consumption. Each sleep transition has energy and delay cost associated to it. The transition time of going into and out of sleep state is denoted as t_{tr}^s and t_{tr}^w respectively. The extra energy consumed during a transition phase is denoted as E_{sw} . The processor has to complete its transition into and out of a sleep state once initiated. When the processor is in sleep state, it has a constant power consumption of P_s . The processor assumed in this model runs at top speed in the active mode and does not support DVFS.

C. Thermal Model

A widely adopted [11], [12] thermal RC model is used to characterise the temperature behaviour of the processor and expressed as a differential equation (**Equation 5**), where C_{th}, R_{th}, P_W, T and T_{amb} are the thermal capacitance (Joule/K), thermal resistance (K/Watts), processor's power consumption (Watts), processor's temperature (K) and the ambient temperature (K) respectively.

$$\frac{dT}{dt} = \frac{1}{C_{th}}P_W - \frac{1}{R_{th}C_{th}}(T - T_{amb}) = \hat{\alpha}P_W - \hat{\beta}(T - T_{amb}) \quad (5)$$

Yang et al. [12] solved the differential equation (**Equation 5**) and derived temperature as a function of time for both active (**Equation 6**) and sleep state (**Equation 7**) modes. The same notations are used here for consistency.

$$T_{act}(\hat{t}, t) = \frac{-(k\theta_1 e^{(\theta_1 - \theta_2)t} + \theta_2)}{a(k e^{(\theta_1 - \theta_2)t} + 1)} \quad (6)$$

$$T_{dor}(\check{t}, t) = (1 - e^{-\hat{\beta}t})\eta + T_{dor}(\check{t}, 0)e^{-\hat{\beta}t} \quad (7)$$

When the processor is in active mode for an interval of $(\hat{t}, \hat{t} + t]$, $T_{act}(\hat{t}, t)$ is the temperature at time instant $\hat{t} + t$ assuming \hat{t} is the time instant in the beginning of execution. Similarly, $T_{dor}(\check{t}, t)$ is a temperature at the end of the interval $(\check{t}, \check{t} + t]$ assuming system in the sleep state starting from a time instant \check{t} . $T_{act}(\hat{t}, 0)$ and $T_{dor}(\check{t}, 0)$ are temperatures at time instance \hat{t} and \check{t} respectively. The parameters $\theta_1 = \frac{b + \sqrt{b^2 - 4ac}}{2}$, $\theta_2 = \frac{b - \sqrt{b^2 - 4ac}}{2}$, $k = \frac{-(aT_{act}(\hat{t}, 0) + \theta_2)}{(aT_{act}(\hat{t}, 0) + \theta_1)}$, $\eta = (T_{amb} + \frac{\hat{\alpha}}{\hat{\beta}}P_s)$, $a = \hat{\alpha}\mathcal{A}$, $b = -\hat{\beta}$ and $c = \hat{\alpha}(P_a + \mathcal{B}) + \hat{\beta}T_{amb}$. Assume, T_{cri} defines the maximum allowed temperature for the safe operation of the chip. The Equation 6 and Equation 7 can be rewritten in terms of temperature and their corresponding equations are given in Equation 8 and Equation 9 respectively. With Equation 8 and Equation 9, one can compute the time units system takes to move from one temperature to another both in active and sleep modes respectively.

$$t_a = \frac{1}{\theta_1 - \theta_2} \ln \left(\frac{-(\theta_2 + T_{act}(\hat{t}, t)a)}{k(\theta_1 + T_{act}(\hat{t}, t)a)} \right) \quad (8)$$

$$t_c = \frac{1}{-\hat{\beta}} \ln \left(\frac{\eta - T_{dor}(\check{t}, t)}{\eta - T_{dor}(\check{t}, 0)} \right) \quad (9)$$

The energy consumption in sleep state for an interval of $[t_1, t_2]$ is $E_s = P_s(t_2 - t_1)$. The active energy consumption E_a is computed by integrating P_a [12] as given in Equation 10.

$$\begin{aligned} E_a &= \int_{t_1}^{t_2} P_a dt = \int_{t_1}^{t_2} (P_{dym} + \mathcal{A}T_{act}(t_1, t_2 - t_1)^2 + \mathcal{B}) dt \\ &= (P_{dym} + \mathcal{B})t \Big|_{t_1}^{t_2} + \frac{\mathcal{A}}{a^2} [\theta_2^2 t + (\theta_1 - \theta_2) \\ &\quad \ln(k e^{(\theta_1 - \theta_2)t} + 1) + \frac{(\theta_1 - \theta_2)}{k e^{(\theta_1 - \theta_2)t} + 1}] \Big|_{t_1}^{t_2} \quad (10) \end{aligned}$$

III. AVAILABLE UTILISATION

The execution of a workload on a processor increases its temperature. When its temperature reaches the thermal threshold, a cooling phase is triggered. The decision should be made about the duration of the cool down phase. Before making such decision, discussion of two conflicting scenarios is required as given below.

- 1) The exponential nature of the thermal model allows the system to perform more execution at high temperatures as the temperature rise in the active phase is slower and the fall in the cooling phase is faster. The leakage current also increases at high temperature and results in additional energy consumption. Moreover, performing execution at high temperatures also increase the number of sleep transitions to decrease its temperature, which is not desirable due to an overhead associated to each sleep transition.
- 2) Conversely, when the processor cools down to low temperatures, its temperature rises faster in the active

phase and falls slower in the cooling phase. The leakage current is also relatively less at low temperatures. Nevertheless, a relatively long cooling phase is required to attain the low temperature. A long cooling phase decreases the system's energy by reduced sleep transitions.

Hence, a trade-off between performance and the energy consumption exists between two different aforementioned cases. In RT systems, the worst-case requirements of the system are known a-priori. Initially the available utilisation of the system is defined as a function of time while later extended to a function of temperature. The **available utilisation** of the system is the maximum amount of execution per unit time that system can ensure respecting the thermal constraint. Assume, T_{max} is the upper threshold temperature after which scheduler switch on the cooling phase. The value of $T_{max} \leq T_{cri}$. The scheduler allows the system to execute unless its temperature reaches T_{max} . Similarly, the cooling phase is switched off when the temperature reaches to a lower threshold temperature $T_o < T_{max}$. The available utilisation U_{avail} of the processor with such repetitive cycles is given in Equation 11, where t_a is the time system takes in active state to reach from T_o to T_{max} and t_c is the time it takes to cool down to T_o from T_{max} .

$$U_{avail} = \frac{t_a}{t_a + t_c} \quad (11)$$

The execution is performed during t_a time interval, while t_c is the idle time. Using the empirical data given in the work of Yang et al. [12], Figure 1 plots the temperature profile of the processor versus time. The cooling phase and the execution phase are exponential functions and the rate of change in temperature is higher in the beginning of their respective phases. This illustrates the fact that one can execute more by setting T_{max} and T_o at high temperatures. The available utilisation of the system for different lengths of execution times in active phase (t_a) are presented in Figure 2. The value of T_{max} is fixed to 400K. Given the system requirements in terms of U_{avail} , one can vary the values of t_a and t_c , to reduce energy consumption while respecting the thermal constraint.

Assume, a system transition into a sleep state in the cooling phase. Equation 8 and Equation 9 can be used to replace the corresponding values of t_a and t_c respectively to define U_{avail} as a function of temperature given in Equation 12. The value of $T_{dor}(\check{t}, 0) = T_{act}(\hat{t}, t)$ and replaced with T_{max} . Similarly, $T_{dor}(\check{t}, t) = T_{act}(\hat{t}, 0)$ and these symbols are replaced with T_o .

$$U_{avail} = \frac{\hat{\beta} \ln \left(\frac{(\theta_1 + T_o a)(\theta_2 + T_{max} a)}{(\theta_2 + T_o a)(\theta_1 + T_{max} a)} \right)}{\hat{\beta} \ln \left(\frac{(\theta_1 + T_o a)(\theta_2 + T_{max} a)}{(\theta_2 + T_o a)(\theta_1 + T_{max} a)} \right) - (\theta_1 - \theta_2) \ln \left(\frac{\eta - T_o}{\eta - T_{max}} \right)} \quad (12)$$

IV. ENERGY CONSUMPTION OF RT SYSTEMS UNDER THERMAL CONSTRAINT

The energy consumption of the system with leakage-aware TCDDPM can be minimised through two different factors.

- 1) Initiating the sleep state for longer intervals to reduce the total cost of sleep transitions and to maximise the idle period in low power state.

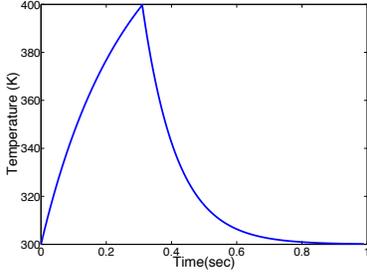


Fig. 1. Temperature Profile

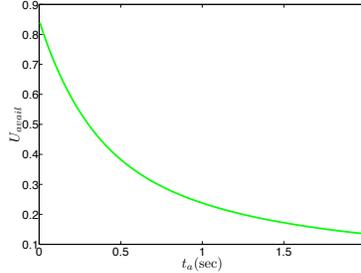


Fig. 2. U_{avail} vs t_a

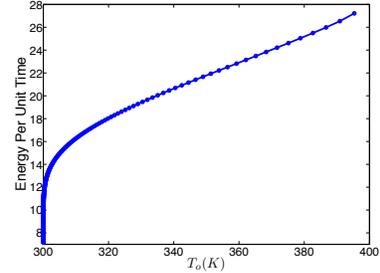


Fig. 3. Energy vs T_o

- 2) Running the system at low operating temperatures to avoid the higher leakage power dissipation at high temperatures.

In the first case, duration of the sleep intervals is increased, the system gets more time to cool down. This effect decreases the available utilisation of the system as the temperature rises at faster rate at low temperatures in active mode and on contrary, the rate of cooling is slower at low temperatures. In the second case, running a system at high temperature increases the leakage power consumption. However, if the operating temperature range, i.e. both T_{max} and T_o , is shifted to low temperatures, the available utilisation of the system also decreases because of the same aforementioned reason. Hence, in both cases the decrease in available utilisation is due to a reduction in the duty cycle.

An optimal solution should consider both factors mentioned above to minimise the overall energy consumption of the system. Nevertheless, intuition is clear that the energy consumption of the system in TCDPM is reduced by running the system at the lowest possible available utilisation (decreasing the duty cycle). One can propose different techniques to find the optimal set of T_{max} and T_o considering both factors for different values of U_{avail} . However, the objective of this research effort is not to find such values, rather to show that idealised DVFS algorithms are equivalent to TCDPM in a sense that both have the same objective to run the system at low available utilisation U_{avail} whenever it is possible. As a first approximation it is assumed that the value U_{avail} is computed by fixing T_{max} to T_{cri} and varying T_o . **Figure 3** shows the energy consumption per unit time (power) of the system using such approximation for different values of T_o ($T_{max} = 400K$). It is evident that the energy consumption of the system increases when increasing the value of T_o . The same fact is also varied by plotting the energy consumption per unit time against the available utilisation in **Figure 4**. It is evident from **Figure 4** that energy consumption increases by running the system at high available utilisation and high utilisation can only be achieved by operating at high temperatures. Given T_{max} and T_o , the values of t_c and t_a can be determined by using **Equation 8** and **Equation 9**.

V. EQUIVALENCE OF IDEALISED DVFS AND TCDPM

The available utilisation U_{avail} given in **Equation 11** provides the execution per unit time for long time intervals (i.e. $\Delta t \gg t_c$), which is virtually equivalent to the normalised speed of the processor. The reduction in the amount of work

per unit time (i.e. available utilisation or virtual speed of the processor) also decreases the energy consumption of the system. This occurs as the amount of work per unit time is decreased by reducing the duty cycle in TCDPM which can be achieved either by allowing the system to stay longer in the sleep state or by decreasing the operating temperature range (i.e. T_{max} and T_o) of the system. This virtual reduction of speed also means prolonging the execution time of the tasks as the temperature rise is exponential and execution per unit of time does not scale linearly with a decrease in temperature. The traditional idealised DVFS theory is also based on a convex function of the power consumption. The decrease in speed/frequency of the processor though saves energy but also prolongs the execution time of the given workload by running the processor slower. In real DVFS, the execution time does not scale linearly with the processor speed $\frac{1}{f}$ (for example, memory access time does not scale with the processor frequency)[13]. However, the above assumption is often made in the literature. Under TCDPM, the execution of the workload is performed at full speed and it behaves almost at 50% speed when given a 50% duty cycle (available utilisation). Similarly, in idealised DVFS, it is assumed the execution scales by a factor of $\frac{1}{f}$. If the frequency is 50%, the execution time scales by a factor of 2 which is equivalent to 50% duty cycle in TCDPM at full speed. Moreover, another reason for similarity is that idealised DVFS has a continuously spectrum of available frequencies and similarly, TCDPM can represent the duty cycle in any ratio. If frequencies are normalised in idealised DVFS, there is a correlation between idealised DVFS frequencies and normalised speed (duty cycle) in TCDPM. In both cases the objective is to reduce the amount of work per unit time to reduce the overall energy consumption of the system.

A. Schedulability Concerns

The similarities between these two problems allow us to apply any existing DVFS algorithms on TCDPM to reduce the energy consumption of the system with some minor modifications in the schedulability analysis and/or speed modifications in TCDPM.

In DVFS, the amount of work per unit time is reduced by decreasing the physical frequency of the processor. The processor runs the instruction at slow but constant rate. The schedulability of the sporadic task model in DVFS is preserved if $\frac{f_t}{f_m} \geq U$, where f_t is the processor's frequency at any time t and f_m is its maximum frequency. On the other side, the

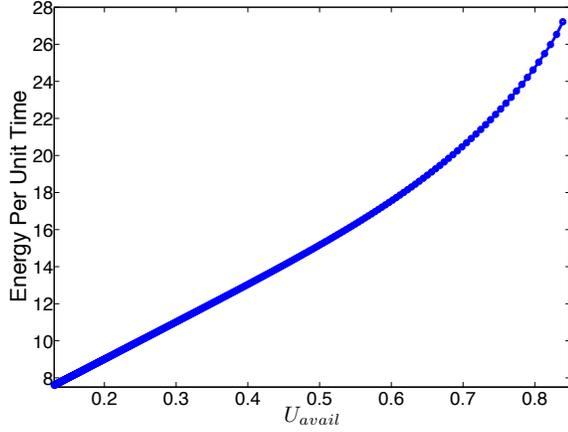


Fig. 4. Energy Consumption vs U_{avail}

suspension of the execution in the cooling phase of TCDPM may cause some of the tasks to miss their deadlines under EDF. Lets consider one task in isolation to show how it can miss its deadline and then propose a method to avoid it. Later in this section, this analysis is extended for multiple sporadic tasks.

1) *Single Task Case:* Figure 5 represents TCDPM processing in an execution vs time graph commonly known as service curve. The continuous line step function represents the ideal-case, where the task starts its execution in the beginning of the active phase. The straight line beneath it shows the gradient of execution i.e. U_{avail} . Assume, a worst-case scenario, i.e. the task arrives in the beginning of the cooling phase and suffers an initial delay of t_c , it may miss its deadline (see dotted step function in Figure 5). This delay reduces the effective amount of work that a system should deliver per unit time to meet all deadlines in the system. Assume t_1 is the initial time instant and t_2 is any time instant such that $t_2 > t_1$ && $t_2 > t_c$. The amount of work done in ideal-case in the interval $\Delta t = t_2 - t_1$ will be equal to $\Delta t U_{avail} = C_x$. While, in worst-case with an initial delay of t_c it will be equal to $U_{avail} \Delta t - U_{avail} t_c = C_y$. By substituting the value of C_x and rearranging, $C_x - C_y = U_{avail} t_c$. This is the maximum delay that a task can have in its T_i .

To preserve the system schedulability, the effect of this additional delay of t_c should be accounted in the requested utilisation. The effect of this error is quantified by computing the requested utilisation U_{req} as given Equation 13. The value of t_c is computed by considering the ideal-case (no blocking in the beginning of execution phase). The scaling of $U_{avail} \geq U_{req}$ ensures that the extra amount of work done per unit time will be greater than or equal to $\frac{t_c}{P_i}$. The schedulability of the single task is ensured if its period satisfy the condition given in Equation 14. Both Equation 13 and Equation 14 are sufficient conditions. Equation 14 computes the number of active phases required to execute the task and adds the corresponding cooling phase, and ensures it is greater than the period/deadline of the task to preserve schedulability.

$$U_{req} = \frac{C_i}{P_i} + \frac{t_c}{P_i} \quad (13)$$

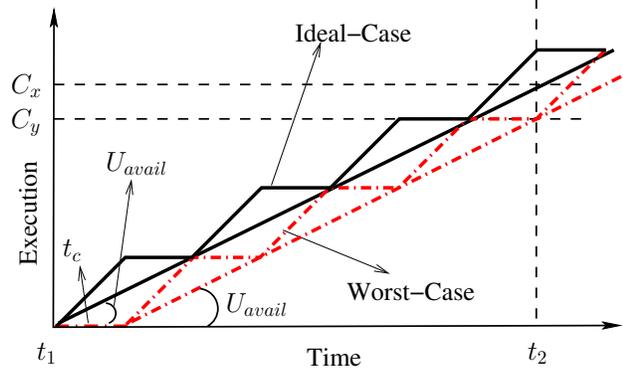


Fig. 5. Service Curve

$$P_i > \left\lfloor \frac{C_i}{t_a} \right\rfloor (t_a + t_c) + (C_i \% t_a) + t_c \quad (14)$$

2) *Multiple Tasks Case:* This analysis is extended to multiple sporadic tasks to ensure their schedulability. First of all, a slight modification is made in U_{req} as given in Equation 15. Instead of $\frac{t_c}{P_i}$, $\frac{t_c}{\min(P_i)}$ is used and now for each period of the highest priority task the amount of extra work will be equal to $U_{avail} t_c$. Similar to a single task case, the value of t_c is obtained by considering the ideal-case and the original value of U_{avail} is raised to U_{req} to ensure the system schedulability of all tasks. Moreover, all the tasks should satisfy the condition given in Equation 16 to check that they are getting enough active phases in their period to compete their execution to ensure the schedulability. The quantisation error that occurs in TCDPM due to cooling and active phases is bounded to $\frac{t_c}{\min(P_i)}$. This is a pessimistic but safe bound. Similar to single task, Equation 15 and Equation 16 are sufficient conditions.

$$U_{req} = \sum_{\forall \tau_i} \frac{C_i}{P_i} + \frac{t_c}{\min(P_i)} \quad (15)$$

$$\forall \tau_i, P_i > \left\lfloor \frac{C_i}{t_a} \right\rfloor (t_a + t_c) + (C_i \% t_a) + t_c \quad (16)$$

Now consider the other effects (that may affect the schedulability of tasks) such as if a task is executing with a worst-case scenario and other tasks are released during its execution. The arriving task may have higher or lower priority when compared to the currently executing task. If there is an arrival of a lower priority task(s) the normal execution of the system is not interrupted at all as it has to wait for the currently running task to complete its execution. Now consider the effect of the higher priority task τ_i . The schedulability of the higher priority task τ_i is ensured by Equation 16. The phasing of τ_i with respect to the phasing of the cooling is of no concern as the overall execution requirement is only increased by C_i . Similarly, it can be shown that by adding extra tasks, the schedulability of the system remains unaffected.

VI. CASE STUDY

This section shows that TCDPM problem can be solved with existing DVFS algorithms. For demonstration purpose,

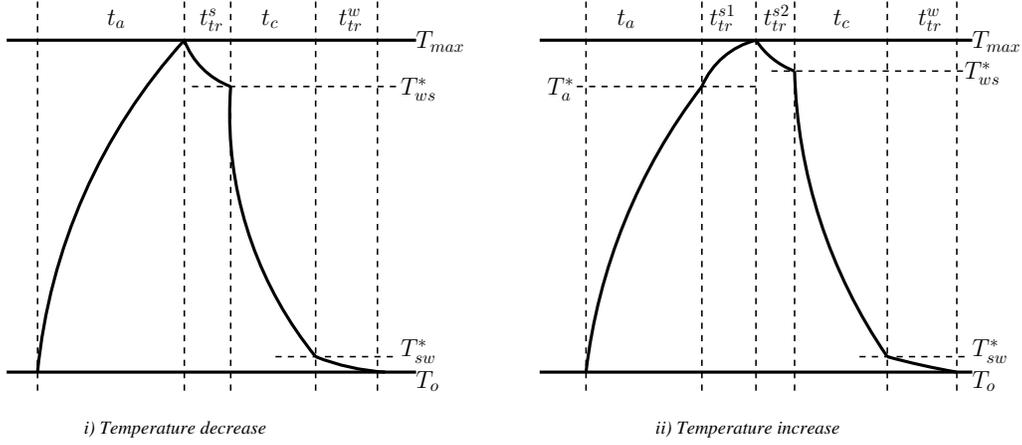


Fig. 6. Temperature Decreases or Increase in Transition Phase

two DVFS algorithms for RT systems from the work of Pillai and Shin [14] are considered in this case study. It is assumed all the frequency set-points of the processor are normalised with the maximum frequency of the processor.

A. Static Allocation of Frequency

In the first algorithm of Pillai and Shin [14], it is assumed that all the tasks execute for their worst-case and they find statically the operating frequency of the processor. The operating frequency f_o of the processor is set to $U \times f_{\max}$. The execution time of all the tasks are scaled by a factor of $\frac{1}{f_o}$. Similarly, U_{avail} in TCDPM corresponds to U in DVFS. The value of U_{req} is computed to eliminate the error caused due to the quantisation effect and set the value of $U_{\text{avail}} \geq U_{\text{req}}$. The selected value of U_{avail} in turn is used to estimate t_a and t_c . Afterwards, periods of all the tasks are checked for condition given in Equation 16

B. Using Generated Slack to further reduce the Frequency

In RT systems, tasks are budgeted for the worst-case scenario but normally, they execute less than their worst-case estimation. The difference of WCET and the actual execution time is collated and terms as execution slack. Pillai and Shin [14] explored execution slack to further reduce the operating frequency. On the early completion of any task the unused execution time is reclaimed and the utilisation of the system is recomputed by considering the actual execution time of the current task. The operating frequency is set accordingly with this newly computed system utilisation. The individual utilisation of the task considering its actual execution is used until its next arrival. On any task arrival, the system utilisation is computed again by replacing the previous individual utilisation of the currently arrived task with $\frac{C_i}{P_i}$. The operating frequency is changed accordingly. This algorithm does the frequency adjustment on the task arrival and on its completion.

Similar to Pillai and Shin's approach [14], TCDPM should also make decisions about changing U_{avail} at the arrival and the completion of all tasks. For the temporal correctness, U_{avail} should be greater than or equal to U_{req} (i.e. $U_{\text{avail}} \geq U_{\text{req}}$). U_{req} is composed of two components. The first component computes the current utilisation of the system, while

second factor considers the effect of blocking. A change in current utilisation of the system will vary the cooling phase, which in turn will effect the blocking time (i.e. second factor in U_{req}). To eliminate this issue, it is assumed that t_c^{\max} is a maximum achievable cooling time in the system. This value can be estimated by setting T_{\max} and T_o to their feasible extremes (i.e. $T_{\max} = T_{\text{cri}}$ and $T_o = T_{\text{amb}}$). In theory the value of t_c^{\max} can reach to infinity if T_o is set equal to T_{amb} . Therefore, for practical purposes T_o can be set to a value $T_{\text{amb}} + t_{th}$, where t_{th} is a small offset to keep t_c^{\max} in a reasonable limit. If $\min(P_i) \gg t_c^{\max}$, then second component in U_{req} equation can be replaced with $\frac{t_c^{\max}}{\min(P_i)}$. Any task in a system cannot suffer from a blocking greater than t_c^{\max} . The first component of U_{req} equation (that estimates the current required utilisation of the system) can be computed in a similar way as computed in Pillai and Shin's approach [14]. However, there is just one exception, if a task arrives in the cooling phase, then system needs to wait for the completion of the current cooling phase to make decision about the new U_{avail} .

C. Reducing Pessimism

The blocking factor of $\frac{t_c^{\max}}{\min(P_i)}$ in U_{req} equation is a pessimistic bound. The tasks rarely face such huge blocking. Another less pessimistic approach is also presented to compute U_{req} . Assume, the previous cooling phase has a length of t_c^{old} . On every task completion or new task arrival in the **active phase**, the individual utilisation U_i of the task is updated and the total system utilisation is recomputed. Considering this new value of total system utilisation, the potential length of the next cooling phase is estimated and denoted as t_c^{new} . The value of U_{req} is set to $\sum_{i=1}^{\ell} U_i + \frac{t_c^{\text{new}}}{\min(P_i)}$. However, if there is a new task τ_i arrival in the **cooling phase** of the system, its processing is postponed by the end of this cooling phase. At the end of the cooling phase, the total system utilisation is computed by considering τ_i 's worst-case execution and the value of t_c^{new} is determined. If t_c^{new} is shorter than the current cooling phase time, then τ_i has suffered an extra delay. To compensate for this extra delay, its individual utilisation U_i is set to $\frac{C_i + \max(t - r_{i,m} - t_c^{\text{new}}, 0)}{P_i}$, where $r_{i,m}$ is the absolute release time of τ_i and t is the current time instant at the end

of cooling phase. With this new value of U_i and t_c^{new} , the value of U_{req} is computed as $U_{req} = \sum_{i=1}^{\ell} U_i + \frac{t_c^{new}}{\min(P_i)} \cdot U_{avail}$ is then set to any feasible value greater than or equal to U_{req} and the corresponding values of t_c and t_a are computed.

One more concern that system needs to deal with is the idle mode. If a system has no workload to execute, it transition into a sleep mode. It is equivalent to the early start of a cooling phase. However, the sleep state is terminated on the arrival of a new task. The delay caused due to this sleep transition can be included in the individual utilisation of the arrived task and that is $U_i = \frac{C_i + t_{tr}^w + t_{tr}^s}{P_i}$. Such additional overhead can be ignored, if the system has an idle mode with zero transition delay to and from active mode. Similar to the examples given in this case study, any other DVFS algorithm can be similarly ported and applied in TCDPM setting.

VII. IMPLEMENTATION CONCERNS

A. Computation of U_{avail} , T_o and T_{max}

In order to reduce the online complexity of the system, an offline table for U_{avail} is computed that contains the corresponding values of T_{max} , T_o , t_a and t_c . Given the values of T_{max} and T_o , the values of t_c and t_a can be easily computed for the required table. The values of T_{max} and T_o against U_{avail} can be computed through various techniques such as exhaustive exploration, dynamic programming, approximation algorithm in which a value of T_{max} is fixed and T_o is varied to get different values of U_{avail} . The values of this table are platform dependent only and are estimated once for the given platform. This table reduces the online complexity of the system to $O(\log_2(n))$ to obtain T_{max} , T_o , t_a and t_c against U_{avail} , where n is the number of U_{avail} entries in the table. The length of this table defines the resolution of U_{avail} . In case of non-linear relation of U_{avail} and the energy consumption, the efficient distribution is to get high resolution of U_{avail} where the rate of change of energy consumption is high.

B. Transition Overheads of the Sleep State

Equation 11 assumes the sleep state has no overhead. However, in reality each sleep transition has a time and energy overhead. The energy overhead comes from the fact that it has to store the current status of the system (e.g. cache write-backs). These overheads may have an impact on the system temperature, which in turn also effect the available utilisation of the system. Lets consider two different cases given below.

1) *Transition Phase Decreases the Temperature:* Consider a case where the the temperature of the system decreases during the transition into a sleep state as shown in **Figure 6-i**. The active state of the system is unaffected. The cooling phase may be affected as the complete circuitry is not off during the transition phase and the cooling time may be different from the cooling behaviour in the sleep state. Therefore, the cooling phase is divided into three intervals as shown in **Figure 6-i**. The curves given in **Figure 6** are arbitrary and their main purpose is to illustrate the differences. In the start of the cooling phase system transition into a sleep state for t_{tr}^s time units and temperature at the end of this transition is T_{ws}^* . The

Parameters	Values
Task-set sizes $ \tau $	{5, 10, 15, ..., 50}
Minimum inter-arrival time P_i for RT tasks	[30ms, 50ms]
Sporadic delay limit $\Gamma \in$	{0, 0.05, 0.1, ..., 1}
Best-case execution time limit C^b	{0.2, 0.25, 0.3, ..., 1}
System Utilisation U	{0.35, 0.40, 0.45, 0.5, ..., 0.7}
Energy Overhead E_{sw} (mJoules)	{10}
$\hat{\alpha}$ (K/Joules)	{26, 27, 28, ..., 35, 35.62}
Dynamic Power P_{dyn} (Watts)	{0.5, 1, 1.5, ..., 5, ..., 9}

TABLE I
OVERVIEW OF SIMULATOR PARAMETERS

sleep state lasts for t_c time units between T_{ws}^* and T_{sw}^* . The value of T_{sw}^* is set such that a system takes t_{tr}^w time units to approach T_o in a transition out phase. The available utilisation in this case can be represented as given in **Equation 17**.

$$U_{avail} = \frac{t_a}{t_a + t_c + t_{tr}^s + t_{tr}^w} \quad (17)$$

2) *Transition Phase Increases the Temperature:* In a case, when the temperature of the system increases during a transition into a sleep state, the execution phase is stopped such that it does not cross T_{max} (**Figure 6-ii**). The transition into a sleep state is divided into two parts. 1) Let t_{tr}^{s1} be the time system takes to do the processing related to the transition of the sleep state (e.g. cache write-backs, IRQ to serve sleep request). 2) t_{tr}^{s2} is the transition time needed to initiate the sleep state after the processing phase and T_{ws}^* is the temperature after t_{tr}^{s2} time units. Assume T_a^* is the system temperature such that if the sleep state is initiated it finishes its sleep related processing before the temperature reaches to T_{max} . In this case, U_{avail} of the system is shown in **Equation 18**.

$$U_{avail} = \frac{t_a + t_{tr}^{s1}}{t_a + t_{tr}^{s1} + t_{tr}^{s2} + t_c + t_{tr}^w} \quad (18)$$

VIII. EVALUATION

The behaviour of the algorithms presented in the case study (**Section VI**) is discussed in this section across different dimensions. The results of idealised DVFS setting of the chosen algorithms presented in the work of Pillai and Shin [14] are compared with the same algorithms ported in TCDPM setting. This comparison shows that the trend of energy saving is consistent demonstrating the equivalence of the idealised DVFS and TCDPM. Note: The result section does not show the direct energy saving comparison of idealised DVFS and TCDPM rather demonstrates the fact that idealised DVFS can be applied in TCDPM setting to save energy. The amount of energy saved in both cases is obviously different and depends on different hardware parameters. The experimental setup used is explained below.

A. Experimental Setup

The discrete event simulator SPARTS (Simulator for Power Aware and Real-Time Systems) [15] has been extended to implement the algorithms discussed in the case study. It is an open source simulator of a real-time device. The variety of

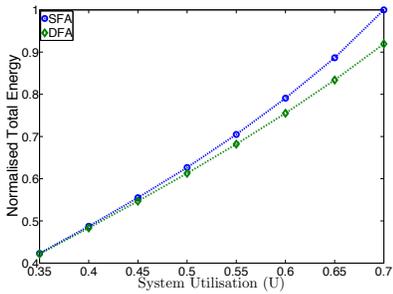


Fig. 7. Effect of Variation in System Utilisation

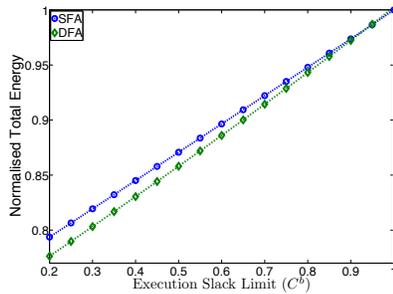


Fig. 8. Effect of Variation in Execution Slack

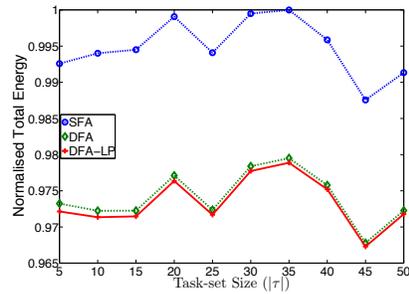


Fig. 9. Effect of Variation in Task-Set Size

applications are considered by varying task-sets size from a large number of fine grained small tasks (50) to a small number of coarse grained tasks (5). SPARTS implements Rate-Based Earliest Deadline first (RBED) framework [16], which provides temporal isolation via enforced budgets. The temporal isolation of the RBED framework allows for mixed-criticality workloads (hard, soft and best-effort type applications). In the context of this research, only hard real-time type applications are considered. The actual individual utilisation per task is generated such that it achieves the target system utilisation. Selecting the utilisation U_i and minimum inter-arrival time P_i for each task according to the limits in Table I, the WCET of each task is computed to be $C_i = U_i \times P_i$.

A two level approach is used for generating a wide variety of different tasks and subsequently varying jobs. In the first level, all tasks are annotated with a limit on the sporadic delay Γ in the interval $[0, \Gamma \times P_i]$ and on best-case execution time (BCET) C_i^b in the interval $[C_i^b \times C_i, C_i]$. Γ models the delay after the minimum-inter arrival time, while, C_i^b gives a limit over the BCET of a task. These two limits are determined for each task. The second level generates different jobs of the same task with varying behaviour dependent on the system state and input parameters. Such varying behaviour is modelled by assigning each $j_{i,m}$ an actual sporadic delay in $[0, \Gamma_i]$ interval and an actual execution time in $[C_i^b, C_i]$ interval. Similar to the analysis, the implicit deadlines $D_i = P_i$ are assumed in the evaluation as well. A uniform distribution is used to generate all the random numbers needed for the given experiment setup. Each task-set is simulated for 100 seconds and each particular configuration is averaged over 100 runs with a seed value of 1 to 100. For more details, see original work of SPART [15].

Various hardware platforms are available in the market with diverse characteristics. The values of $\hat{\beta}$, P_s , $t_{tr}^s = t_{tr}^w$, \mathcal{A} , \mathcal{B} , E_{sw} , T_{amb} and T_{max} are adopted from [12] and are fixed to 9.52/ sec, 50 μ Watt, 5 msec, 0.0002188 Watt/ K^2 , 8.5143 Watt, 10 mJoules, 300 K and 373 K respectively. Other parameters such as P_{dyn} , $\hat{\alpha}$ are considered as a variable in different set of experiments to vary the hardware platform behaviour. SPARTS simulator is used with the configurations summarised in Table I. The default values are underlined.

B. Results

In this section, all the results presented below are normalised to the highest value in the corresponding graph. The default value of the parameter is considered if not mentioned in the

description of the experiment. The static frequency allocation algorithm (Section VI-A), dynamic frequency allocation algorithm (Section VI-B) and dynamic frequency allocation algorithm with reduced pessimism (Section VI-C) are labelled as *SFA*, *DFA* and *DFA-LP* respectively. As the results section frequently refers to the simulations results of Pillai and Shin [14], therefore, it is important to mention that *SFA* and *DFA* in an idealised DVFS setting are termed as static EDF (*staticEDF*) and cycle-conservative EDF (*ccEDF*) respectively. It has been observed that the difference of overall energy consumption between *DFA* and *DFA-LP* is negligible with few exceptions. Therefore, only the comparison of *SFA* and *DFA* is presented, while *DFA-LP* is mentioned only when it makes difference compared to *DFA*.

Initially, the effect of change in the system utilisation is studied for three different approaches. Figure 7 presents the results for a task-set size of 10 with $\Gamma = 0$ and $C^b = 0.2$. The increase in system utilisation obviously increases the energy consumption of the system. This trend is consistent with the results of Pillai and Shin [14], where, the energy consumption of *staticEDF* and *ccEDF* also increases with the system utilisation. The interesting fact evident from Figure 7 is the difference of energy consumption of *SFA* and *DFA*. *SFA* only exploits the spare capacity available in the system schedule called static slack that occurs as the system is loaded less than what can be guaranteed by the schedulability tests and computes U_{avail} considering WCET time of the tasks. On the other hand, *DFA* makes use of the static slack in the offline phase, and utilises the execution slack ($C_i - c_{i,m}$) when recomputing U_{avail} online on early completion of all tasks. The execution slack helps to further reduce the energy consumption by decreasing the demand on U_{avail} . The same reason explains the difference of energy consumption at a utilisation of $U = 1$. *DFA* consumes approximately 9% less energy when compared to *SFA*. Furthermore, *SFA* and *DFA* behave identical if it is assumed that all the task execute for their WCET. The same is true for *staticEDF* and *ccEDF*.

The effect of variation in the execution slack available in the schedule is analysed in Figure 8 with $\Gamma = 0$, $U = 0.5$ and task-set size of 10. The value of C^b is varied from 0.2 to 1. The increase in the value of C^b means a decrease in the execution slack. At $C^b = 1$, the amount of execution slack becomes zero as all tasks execute for their WCET. Therefore, at such utilisation ($U = 1$) both algorithms (*SFA* and *DFA*) perform identical due to unavailability of any source of slack

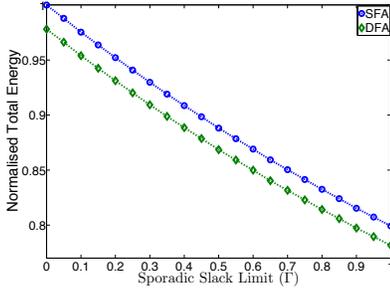


Fig. 10. Effect of Variation in Sporadic Slack

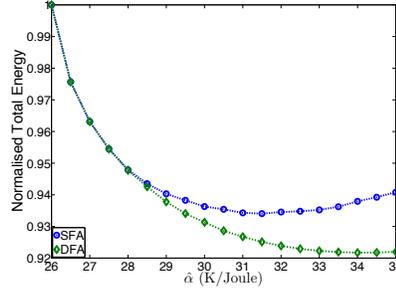


Fig. 11. Effect of Variation in $\hat{\alpha}$

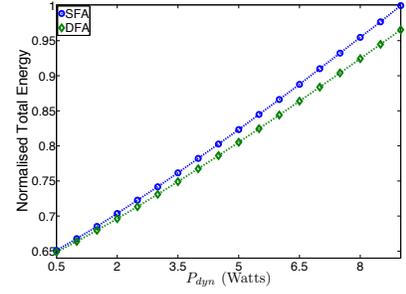


Fig. 12. Effect of Variation in Dynamic Power

and the same observation holds for *staticEDF* and *ccEDF*. In general, the energy consumption of the system increases with a decrease in the amount of execution slack in schedule. The similar trend is followed by the energy consumption of *staticEDF* and *ccEDF* in the idealised DVFS that verifies the equivalence. The *DFA* algorithm exploits execution slack in the system by adapting U_{avail} to a low value according to the system requirement. Nevertheless, *SFA* only makes use of the execution slack by initiating early start of cooling phase and keeping the system in a sleep mode till the arrival of new task. However, it does not readjust U_{avail} to decrease the energy consumption of the system. Likewise to *DFA*, *ccEDF* can reduce the system frequency with execution slack to save the energy. The energy consumption of *staticEDF* is not affected with a variation in the execution slack as Pillai and Shin [14] assumed in their experimental setup that energy consumption of the system is zero in the idle phase with negligible transition overhead. The earlier start of cooling phase helps to save energy in *SFA*.

The effect of task-set size is analysed in Figure 9. The *DFA-LP* algorithm is also included in this experiment for comparison. The task-set size is varied from 5 to 50. This experiment shows the difference of energy consumption does not depend on the task-set size as the difference is negligible. For instance, the maximum difference of energy consumption exists between a task-set size of 35 and 45, and is equal to $\approx 1\%$. The main factors are the system utilisation and the amount of dynamic slack in the system. The *staticEDF* and *ccEDF* also behave similar for different task-set sizes.

This work also analyses extra task-set property called sporadic delay, which is not investigated in the work of Pillai and Shin [14] due to period task model. In the analysis, the system is analysed with another worst-case assumption that each job of a sporadic task will be released as soon as possible i.e. released periodically with the minimum inter-arrival time but it rarely occurs in HRT system. The extra spare time generated due to sporadic delays is termed as Sporadic Slack. Figure 10 demonstrate the behaviour of the system towards sporadic slack in the system. The task-set size, C^b and U are fixed to default values. Γ is varied from 0 to 1 with a step size of 0.05. The increase in the value of Γ increases the sporadic slack. Normally, it is impossible to determine sporadic slack beforehand. Therefore, *SFA* and *DFA* make use of such slack in the cooling phase by extending it till next task arrives. The behaviour of all the algorithms is approximately linear

with the variation of Γ . The energy consumption of the system reduces to 20% from $\Gamma = 0$ (no sporadic slack) to $\Gamma = 1$. This experiment demonstrates that the proposed algorithms can exploit this type of slack implicitly.

After analysing the task-level properties, the hardware platform specific parameters ($\hat{\alpha}$ and P_{dyn}) are considered. The hardware platform specific parameters $\hat{\alpha}$ presented here is not evaluated by Pillai and Shin [14] as it is specific to the temperature model and not relevant in DVFS case. This parameter is discussed to demonstrate the behaviour of the TCDPM algorithms for different types of hardware platforms.

$\hat{\alpha}$ is the inverse of thermal capacitance C_{th} and has a unit of $K/Joules$. The higher value of $\hat{\alpha}$ heats up the hardware platform quickly but cooling phase is independent of it. Figure 11 shows the variation in the energy consumption with different values of $\hat{\alpha}$. The values of P_{dyn} and E_{sw} are fixed to 5 Watts and 0.01 Joules respectively. As the system heats up at a fast rate for a large value of $\hat{\alpha}$, therefore, the active phase of the system shortens with an increase in the value of $\hat{\alpha}$. This decreases the leakage power consumption of the system, as the system stays for a shorter period of time at high temperatures. On contrary, a low value of $\hat{\alpha}$ has long active period and it takes longer to heat up the system. System though executes more but consumes more leakage energy. Therefore, Figure 11 shows that with an increase in the value of $\hat{\alpha}$ the energy consumption of the system decreases. The difference in energy consumption of *SFA* and *DFA* increases at high values of $\hat{\alpha}$. This is motivated by the fact that when system heats up quickly it is important to exploit the available slack consciously. On one side, the system's active phase is shorter and hence, its leakage power consumption reduces. On the other side, the decrease in active phase time means system needs to operate at high U_{avail} values, i.e. short cooling phase. Hence, the effect of execution slack becomes important to extend the cooling phase by reducing the duty cycle i.e. U_{avail} .

The thermal behaviour of the system is modified by increasing the dynamic power consumption of the system. Both leakage and dynamic power consumption contribute to the temperature increase. Varying the dynamic power consumption of the system, not only changes the thermal behaviour but also varies the ratio of dynamic to leakage power consumption. The dynamic power consumption of the system is varied from 0.5 Watts to 9 Watts. The leakage power consumption of the system at ambient temperature is 11.178 Watts. Therefore, the ratio of P_{dyn} to P_{lkg} varies between 0.0447 to 0.8052.

The resulting energy consumption of the system is presented in Figure 12. The increase of dynamic power increases the energy consumption. There is $\approx 35\%$ rise in energy by varying P_{dyn} from 0.5 Watts to 9 Watts. Furthermore, the difference in energy consumption of *SFA* and *DFA* also increases with an increase in dynamic power. The increase in dynamic power increases the temperature of the system and it heats up at a fast rate. Therefore, the active phase of the system decreases that consequently decreases the length of cooling phase. The leakage power consumption parameters are not altered, hence, it varies proportionally and does not affect the behaviour. The shortening of active and cooling phase enhances the need for effective management of U_{avail} , which is obviously better in *DFA* when compared to *SFA*. Therefore, *DFA* performs better at high dynamic power consumption. A similar parameter called idle level (ratio of energy consumption in idle cycle to energy consumption in active cycle) is explored by Pillai and Shin [14]. If the power consumption of the system in idle mode is considered constant, then idle level essentially means a variation in dynamic power. The increase in idle level mean decrease in dynamic power and vice versa. Pillai and Shin [14] showed that a decrease in the idle level increases the energy consumption of *staticEDF* and *ccEDF*. This is consistent with the results presented in Figure 12 that shows that an increase in dynamic power (decrease in idle level) increases the energy consumption of *SFA* and *DFA*.

Each sleep transition has an energy overhead associated to it, which is modelled as E_{sw} in the given system model. The frequent sleep transitions are undesirable and increase the energy consumption. The number of sleep transitions of the previous experiments is shown in Figure 13. In this experiment, the normalised sleep transitions of *DFA-LP* are also included in the comparison as well. An increase in dynamic power heats up the system quickly and decreases the active phase, which in turn also increases the number of sleep transitions. This effect is evident from Figure 13 that shows an increase in the number of sleep transition with an rise in dynamic power consumption. As mentioned previously, *DFA* and also *DFA-LP* manage U_{avail} effectively and extend their cooling phase, hence their number of sleep transition are fewer when compared to *SFA*. Especially at high dynamic power consumption, this difference reaches to $\approx 22\%$. Furthermore, the difference of *DFA* and *DFA-LP* is also visible in this experiment at high value of dynamic power consumption that shows the minor gains of the proposed optimisation.

IX. CONCLUSIONS AND FUTURE DIRECTIONS

In this research effort, it is demonstrated that idealised DVFS and TCDPM are very similar in their nature and with some minor modifications in the schedulability analysis and online mechanisms, the work done for DVFS algorithms can be applied to TCDPM to save energy. This strategy allows to relax the assumptions commonly made in the literature (such as frame based RT system, single task, neglecting energy and temperature independent leakage power consumption) of TCDPM and to apply it on generic RT task model under

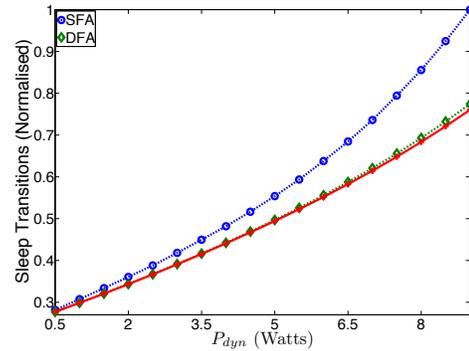


Fig. 13. Number of Sleep Transitions

dynamic priorities. This work has shown the proof of the concept with the help of the case study on DVFS algorithms of the literature. The performance of DVFS algorithms applied to TCDPM is investigated for a broad range of workloads and set of different hardware platforms. The results show the effectiveness of the proposed algorithms. In future, it is intended to look into the efficient ways to find the values of T_o and T_{max} against U_{avail} . It will be interesting to explore the optimal method to achieve such values. This algorithm will be extended to multicore platform implementing RT systems.

ACKNOWLEDGEMENTS

This work was partially supported by National Funds through FCT (Portuguese Foundation for Science and Technology) and by ERDF (European Regional Development Fund) through COMPETE (Operational Programme 'Thematic Factors of Competitiveness'), within [REPOMUC] project, ref. FCOMP-01-0124-FEDER-015050, and by ESF (European Social Fund) through POPH (Portuguese Human Potential Operational Program), under PhD grant SFRH/BD/70701/2010.

REFERENCES

- [1] V. Tiwari, D. Singh, S. Rajgopal, G. Mehta, R. Patel, and F. Baez, "Reducing power in high-performance microprocessors," in *Proceedings of the 36th Design Automation Conference*, June 1998, pp. 732–737.
- [2] J. Kong, S. W. Chung, and K. Skadron, "Recent thermal management techniques for microprocessors," *ACM Computing Surveys*, vol. 44, no. 3, Jun. 2012.
- [3] N. Bansal, T. Kimbrel, and K. Pruhs, "Speed scaling to manage energy and temperature," *Journal of the ACM*, vol. 54, no. 1, pp. 3:1–3:39, Mar. 2007.
- [4] J.-J. Chen, C.-M. Hung, and T.-W. Kuo, "On the minimization of the instantaneous temperature for periodic real-time tasks," in *Proceedings of the 13th IEEE Real-Time and Embedded Technology and Applications Symposium*, April 2007, pp. 236–248.
- [5] V. Chaturvedi and G. Quan, "Leakage conscious dvs scheduling for peak temperature minimization," in *Proceedings of the 16th Asia and South Pacific Design Automation Conference*, 2011.
- [6] V. Chaturvedi, H. Huang, and G. Quan, "Leakage aware scheduling on maximum temperature minimization for periodic hard real-time systems," in *10th CIT*, July, 2010, pp. 1802–1809.
- [7] J.-J. Chen, S. Wang, and L. Thiele, "Proactive speed scheduling for real-time tasks under thermal constraints," in *Proceedings of the 15th IEEE Real-Time and Embedded Technology and Applications Symposium*, 2009.
- [8] G. Quan and V. Chaturvedi, "Feasibility analysis for temperature constraint hard rt periodic tasks," *IEEE Transactions on Industrial Informatics*, 2010.

- [9] H. Huang and G. Quan, "Leakage aware energy minimization for real-time systems under the maximum temperature constraint," in Proceedings of the 48th ACM/IEEE Conference on Design Automation Conference, 2011, pp. 479–484.
- [10] W. Liao, L. He, and K. Lepak, "Temperature and supply voltage aware performance and power modeling at microarchitecture level," IEEE Transactions on CAD ICAS, vol. 24, no. 7, pp. 1042 – 1053, july 2005.
- [11] L. Yuan, S. Leventhal, and G. Qu, "Temperature-aware leakage minimization technique for real-time systems," in Proceedings of the International Conference on Computer Aided Design, 2006.
- [12] C.-Y. Yang, J.-J. Chen, L. Thiele, and T.-W. Kuo, "Energy-efficient real-time task scheduling with temperature-dependent leakage," in Proceedings of the 47th ACM/IEEE Conference on Design Automation Conference, 2010, pp. 9–14.
- [13] D. C. Snowdon, S. M. Petters, and G. Heiser, "Accurate on-line prediction of processor and memory energy usage under voltage scaling," in Proceedings of the 7th International Conference on Embedded Software, Salzburg, Austria, Oct. 2007, pp. 84–93.
- [14] P. Pillai and K. G. Shin, "Real-time dynamic voltage scaling for low-power embedded operating systems," in Proceedings of the 18th ACM Symposium on Operating Systems Principles, Oct. 2001.
- [15] B. Nikolic, M. A. Awan, and S. M. Petters, "SPARTS: Simulator for power aware and real-time systems," in Proceedings of the 8th IEEE International Conference on Embedded Software and Systems, Changsha, China: IEEE, Nov. 2011, pp. 999–1004.
- [16] S. A. Brandt, S. Banachowski, C. Lin, and T. Bisson, "Dynamic integrated scheduling of hard real-time, soft real-time and non-real-time processes," in Proceedings of the 24th IEEE Real-Time Systems Symposium, Cancun, Mexico, Dec. 2003, p. 396.