



Thermal-aware resource allocation in earliest deadline first using fluid scheduling

International Journal of Distributed
Sensor Networks
2018, Vol. 15(3)
© The Author(s) 2019
DOI: 10.1177/1550147719834417
journals.sagepub.com/home/dsn


Muhammad Naeem Shehzad¹, Qaisar Bashir¹, Ghufraan Ahmad²,
Adeel Anjum² , Muhammad Naeem Awais¹, Umar Manzoor³,
Zeeshan Azmat Shaikh⁴, Muhammad A Balubaid⁵ and Tanzila Saba⁶

Abstract

Thermal issues in microprocessors have become a major design constraint because of their adverse effects on the reliability, performance and cost of the system. This article proposes an improvement in earliest deadline first, a uni-processor scheduling algorithm, without compromising its optimality in order to reduce the thermal peaks and variations. This is done by introducing a factor of fairness to earliest deadline first algorithm, which introduces idle intervals during execution and allows uniform distribution of workload over the time. The technique notably lowers the number of context switches when compare with the previous thermal-aware scheduling algorithm based on the same amount of fairness. Although, the algorithm is proposed for uni-processor environment, it is also applicable to partitioned scheduling in multi-processor environment, which primarily converts the multi-processor scheduling problem to a set of uni-processor scheduling problem and thereafter uses a uni-processor scheduling technique for scheduling. The simulation results show that the proposed approach reduces up to 5% of the temperature peaks and variations in a uni-processor environment while reduces up to 7% and 6% of the temperature spatial gradient and the average temperature in multi-processor environment, respectively.

Keywords

Embedded systems, fluid scheduling, thermal-aware scheduling, simulation, multi-core systems

Date received: 16 November 2018; accepted: 6 February 2019

Handling Editor: Suleman Khan

Introduction

With the innovation in complementary metal oxide semiconductor (CMOS) technology, the size of the chip is constantly decreasing. However, the operating voltages of circuit are not decreasing at the same proportion. It has resulted in elevated chip power densities and high temperatures. High chip temperatures not only challenge the system's reliability but also increase the total power consumption.¹ Hotspots (localized increase of temperature of any particular core due to concentrated on-chip computation activity) and thermal cycles (temperature variations on a chip between upper and lower bounds) are two common thermal

¹Department of Electrical and Computer Engineering, COMSATS University Islamabad, Lahore, Pakistan

²COMSATS University Islamabad, Islamabad, Pakistan

³Tulane University, New Orleans, LA, USA

⁴University of the Punjab, Lahore, Pakistan

⁵Department of Industrial Engineering, Faculty of Engineering, King Abdulaziz University, Jeddah, Saudi Arabia

⁶Prince Sultan University, Riyadh, Saudi Arabia

Corresponding author:

Muhammad Naeem Shehzad, Department of Electrical and Computer Engineering, COMSATS University Islamabad, Lahore Campus, Lahore 54000, Pakistan.

Email: naeem.shehzad@cuilahore.edu.pk



Creative Commons CC BY: This article is distributed under the terms of the Creative Commons Attribution 4.0 License

(<http://www.creativecommons.org/licenses/by/4.0/>) which permits any use, reproduction and distribution of the work without

further permission provided the original work is attributed as specified on the SAGE and Open Access pages (<https://us.sagepub.com/en-us/nam/open-access-at-sage>).

issues. Hotspots can cause problems like electro-migration, dielectric breakdown and ultimately permanent device failure while thermal cycles cause accelerated package fatigue and material deformation.² Moreover, the costly packaging solutions and cooling mechanisms required to diminish these effects increase the cost of the system. The power and energy management is even more crucial in embedded systems, since they are mostly battery operated with few cooling mechanisms. Moreover, tremendous growth of Internet of things (IoTs) in recent years and the respective role of embedded system applications have increased the importance of optimal utilization of processing resources in the constrained environments. This is especially critical in case of real-time applications where the processing is strictly time-constrained.

Sensing the sensitivity of the problem, numerous temperature-aware management techniques have been proposed in the previous work. These techniques are either based on hardware approach or software approach or a combination of the two. Some state-of-art scheduling-based thermal management techniques have been highlighted here.

Dynamic power management (DPM)³ and dynamic voltage and frequency scaling (DVFS)⁴ are commonly used techniques to solve the thermal as well as power issues at operating system level. DPM reduces the static power by putting the processor in low power state (sleep mode, deep sleep mode etc.) during idle time depending on workload. DVFS is used where the processor has an ability to operate at different pairs of voltage and frequency to minimize the power usage while still guaranteeing the deadlines. By operating the device at lower voltage and frequency, the dynamic component of the power is reduced. Numerous research works have targeted the modification of DPM and DVFS techniques to optimize the temperature. Lee et al.⁵ present a thermal control solution based on DVFS. The scaling of voltage and frequency is done on the prediction of local chip temperature. Baati and Auguin⁶ use a DVFS-DPM-based temperature-aware technique, using some heuristics, to address the problem of both thermal peaks and thermal variations at the same time. Mahmood et al.⁷ propose a combination of genetic algorithm (GA) and stochastic evolution algorithm to allocate and schedule real-time tasks over a DVFS-enabled processor to reduce energy and power. Nogues et al.⁸ introduce a framework to design energy and temperature efficient systems. This framework allows the adaption of both DVFS and DPM according to the requirements of the application.

Scheduling based on load balancing is a very popular approach, where the tasks are shifted to relatively less busy cores to avoid temperature peaks.⁹ Kursun and Cher¹⁰ introduce a temperature-estimation technique for multi-processor systems using already

available thermal sensors. The technique reduces the thermal heating without degradation of performance. The technique reduces the thermal imbalances between the cores by using the on-chip thermal sensor's generated temperature variation map in multi-core system. The information is used to address the variation in temperature at run time. It lowers the overall temperature by 4.5°C. Salamy¹¹ presents a temperature-aware scheduling technique for multi-core embedded systems. The technique uses an optimal integer linear programming (ILP) formulation and sub-optimal solutions based on a GA. Merkel et al.¹² present a temperature-aware scheduling that balances the energy consumption among the cores in order to avoid the thermal gradients in the multi-core system. The technique observes the nature of the tasks, and the workload is shifted on the bases nature of tasks to avoid processor throttling. Bashir et al.¹³ present a thermal-aware technique using the principal of load balancing that addresses the thermal emergencies while reducing the switching time between the cores. This is done by estimating the temperature patterns based on the recorded thermal history of similar task sets. However, this is a static technique and cannot be used for an unknown set of tasks. The authors modified this static method into a dynamic technique where the temperature is measured at run time.¹⁴ This online temperature measurement is consequently used for prediction of temperature for reducing the switching time as well as the thermal emergencies.

Due to increasing use of heterogeneous architecture in multi-processor systems, a number of researchers have addressed the thermal issues in such systems. Zhou et al.¹⁵ present a thermal-aware task-scheduling technique for energy minimization in heterogeneous multi-processor system on chip. The first stage of this technique analyses the energy optimality, and the second stage investigates the peak temperature and leakage power. Alsubaihi and Gaudiot¹⁶ propose a thermal and power-aware technique named PETRAS for heterogeneous environment. It improves the system performance by considering task mapping, core utilization, and threads allocation in the scheduling policy. Alsafrjalani and Adegbija¹⁷ present a temperature- and energy-aware scheduling algorithm names TaSaT for heterogeneous and re-configurable hardware. The algorithm reduces the thermal issue without any prior knowledge of the task set without compromising the performance. The technique allows an intelligent selection of available cores to reduce static and dynamic power.

A number of power-aware techniques have been proposed for directed acyclic graph (DAG)-based applications. An energy-efficient scheduling approach on multi-core systems for real-time tasks is proposed by Guo et al.¹⁸ It efficiently assigns the workload and

controls the execution pattern for economizing the power without violating any deadline. The authors also extended the same concepts on sporadic DAG-based tasks with implicit deadline.¹⁹ The technique reduces the power consumption by more than 60% when compare to the simple DAG-based schedulers.

Iranfar et al.²⁰ propose a scheduling technique to avoid thermal emergencies in multi-core systems. In this approach, core consolidation and deconsolidation is performed by considering power and peak temperature. These constraints are also used to find optimal voltage and frequency settings. It improves the mean time to failure of device when compared to state-of-the-art techniques by 47%. Ahmed et al.²¹ present the idea of task set characterization on the basis of their thermal utilization. They consider the increase in temperature due to the execution of a task set and use this information for providing a solution that does not cause any thermal violations in a uni-processor environment. Thereafter, they use speed-scaling technique in order to minimize the system temperature. In the extension of their work, Ahmed et al.²² derive the sufficient and necessary condition for thermal feasibility on a single-core system and extended some results on a multi-core system. They proposed that if the computational and thermal utilization is less than or equal to 1, then a schedule is possible where all the tasks meet their deadlines without avoiding temperature thresholds using a global positioning system (GPS)-inspired fluid-scheduling algorithm. They used the PFair algorithm²³ for scheduling of tasks on a single core while ignoring the pre-emptions overhead. PFair is an optimal multi-processor algorithm based on fairness; however, it incurs a lot of overhead due to the task migrations and context switches. Although Ahmed et al.²² used PFair algorithm in the uni-processor environment, where no migration issues are faced, but still it causes a lot of context switches due to the strict fairness limits applied to each task.

In this article, we address the problem of scheduling for a set of real-time periodic tasks over a single and multi-processor system while minimizing the temperature peaks and variations without missing the deadlines. To accomplish it, we propose a modification in earliest deadline first (EDF), the most successful optimal algorithm in uni-processor environment to address the thermal issues without disturbing the optimality of the algorithm. This is done by introducing a factor of fairness in the algorithm. The proposed algorithm achieves the same level of fairness as proposed by Ahmed et al.;²² however, it significantly lowers the task switching. Although the technique is proposed for the uni-processor platform, the results can eventually be used in partitioned scheduling algorithms for the multi-processor environment. Partitioned scheduling converts the multi-processor scheduling problem in multiple uni-

processor scheduling problems and then solve it using a uni-processor scheduling technique.

The main contribution and significance of this work are as follows:

- The proposed technique reduces the temperature peaks, temperature temporal and spatial gradients in the multi-core system.
- The proposed technique incurs less computational overhead than the techniques based on the same amount of fairness.
- The proposed technique does not require any specific hardware for implementation and thus can be used in different type of computational resources which are quite likely in IoTs.

The rest of the article is organized as follows. Section ‘System model’ presents the system and power model. Section ‘Relevant terminologies’ describes necessary terminologies while the proposed Fair-EDF algorithm is discussed in section ‘Fair-EDF’. Section ‘Results’ discusses the simulation setup and results. Concluding remarks are outlined in section ‘Conclusion’.

System model

Task model

We consider a set τ , composed of N independent, synchronous and periodic tasks, to be scheduled on a single processor. The tasks have implicit deadlines, that is, the time period is equal to the deadline. Each task T_i is characterized by a worst case execution time C_i and a time period P_i . It means that the arrival of two successive jobs of T_i is exactly separated by P_i time units, and each job must be executed for C_i time units before the next job arrives. The utilization factor of T_i is defined as $u_i = C_i/P_i$. U is regarded as the total utilization factor of the system and is equal to the sum of individual utilization factor of all the tasks in the system as expressed in equation (1)

$$U = \sum_{i=1}^N u_i \quad (1)$$

Power model

The power model by Jejurikar et al.²⁴ is used in this research work. The power of each core in this model is expressed by equation (2)

$$P_{Dynamic} + P_{Static} \quad (2)$$

The dynamic part of the power is the function of operating frequency and the voltage used in the system and is given by equation (3)

$$P_{Dynamic} = C * F * V^2 \quad (3)$$

where C represents the switching capacitance, F is the switching frequency, and V is the supply voltage. The static component of power is also termed as leakage power. The leakage power is significantly increased by high temperature. The effect of temperature on leakage power is expressed in equations (4) and (5)

$$P_{Static} = I_L * V \quad (4)$$

where I_L is the leakage current, I_L is a function of temperature and is expressed in equation (5)

$$I_L = I_O * \frac{T^2}{T_O^2} * e^{(a*V*(T-T_O)/(T*T_O))} \quad (5)$$

where a is the switching capacitance, T is the temperature of the operation; I_O is the leakage current at reference temperature T_O . Leakage current has an exponential relation with the temperature that results in an exponential increase in the static power. Moreover, the total power consumption of system also is enhanced. Therefore, the chip temperature is directly affected by variation in ambient temperature.

Energy consumption on any given period of time [g , h] can be calculated by using equation (6)

$$E = \int_g^h p(t) dt \quad (6)$$

Thermal model

We have used the thermal model of the chip as defined by analytical model of temperature in microprocessors (ATMIs).²⁵ Like most other temperature models, ATMI is a linear temperature model. It defines two layers of chip, one of silicon and the other layer consists of $L \times L$ metal dimension. User must define the necessary parameters including the dimension of each layer, their thermal conductance and value of their thermal diffusivity. The parameters corresponding to silicon and metal layers are also set. The impact of edges and interface material between layers is not considered in ATMI. It is modelled by conductance as expressed in equation (7)

$$h_i = \frac{k_i}{d_i} \quad (7)$$

where k_i and h_i denote the thermal conductivity and conductance of interface material, respectively, while d_i denotes the measured thickness of the interface. The ambient temperature is assumed to be uniform. The ambient temperature is not the part of the ATMI model and is added to get absolute value of temperature.

ATMI uses core functions to get transient temperature response against power and steady-state response is obtained by applying principle of superposition. Heat equations are introduced to model steady-state and transient temperatures

$$\frac{dT}{dt} = b * (T_S - T) \quad (8)$$

where b is specific value of thermal parameter which is dependent on hardware, and T_S is the steady-state temperature. By solving equation (8) for $T_{(0)} = T_{AMBIENT}$ and $T(\infty) = T_S$

$$T(t) = T_S - (T_S - T_{AMBIENT}) * e^{-bt} \quad (9)$$

After the calculation of thermal parameters, the temperature of each core is estimated in the model based on the principle of superposition. We can notice that the thermal behaviour of each core is different from that of other cores depending on application task set. Furthermore, to limit the temperature from exceeding a certain temperature, the feature of throttling thermal mechanism is also included in ATMI.

Relevant terminologies

The following concepts will be helpful to understand the proposed algorithm.

Server

A server S is a virtual task in the system with maximum utilization factor U_S of 1. The concept of the server was first introduced by Regnier et al.²⁶ It comprises the real tasks as its clients. When a server is running, actually one of its client tasks is using the processor. The utilization factor of server U_S is sum of individual utilization factor of the clients. If there are n client tasks in a server then utilization factor of the server U_S is given by the following equation (10)

$$U_S = \sum_{i=1}^n u_i \quad (10)$$

Fairness

The concept of fairness is basically derived from the idea of fluid schedule. The core idea of fairness is to enforce task execution by controlling the deviation from the (ideal) fluid schedule. Such deviation is measured for a task by a parameter lag which is equivalent to an allocation error²³.

Definition (fluid schedule): a scheduling plan is said to be fluid if at any time $t \geq 0$, a task T_i has been executed for exactly $u_i \times t$ units of time.

Definition (lag): The lag of task T_i at time t is the difference between the amount of time units actually executed by T_i until time t (i.e. $C_i(t)$) and the amount of time units it would have executed in the fluid schedule by the same instant t

$$\text{lag}(T_i, t) = u_i * t - \sum_{l=0}^{t-1} X(T_i, l) \quad (11)$$

where $u_i * t$ represents the total executed time units of task T_i at time t in an ideal system. X is the actual schedule function defined for each slot of time. $X(T_i, l) = 1$, which means that task is scheduled and is executed in that slot l while $X(T_i, l) = 0$ employs that task is not scheduled over the slot. $\sum_{l=0}^{t-1} X(T_i, l)$ is the total execution of task T_i at time t in the real sequence.

The fairness can be varied by relaxing or tightening the value of lag as per requirement.

Fair-EDF

Fair-EDF is an improved form of EDF algorithm. It works in two steps:

- The execution of task set in Fair-EDF is considered as execution of server task S . Since the EDF is an optimal uni-processor algorithm, the maximum load cannot be higher than the capacity of a single processor. Hence, only one server is required for the whole task set on a single processor.
- The condition of lag is applied to utilization factor of the server U_S at each time unit during the execution. If U_S is the total utilization factor of a server, the value of the lag of the server can be written in the form of following equation

$$\text{lag}(S, t) = U_S * t - \sum_{l=0}^{t-1} X(S, l) \quad (12)$$

where $\sum_{l=0}^{t-1} X(S, l)$ are the total executed units of the server at time t .

Definition (Fair-EDF): a schedule is called Fair-EDF if at any time $t \geq 0$, for a server S , $|\text{lag}(S, t)| < 1$.

The introduction of fairness by controlling the value of lag in Fair-EDF results in uniform distribution of workload over time which lowers the temperature peaks and temperature variations at the same time. Each task in the task set executes at approximately uniform rate inside the umbrella of a server. This causes the Fair-EDF to work as non-work-conserving algorithm in different to that of EDF which is work-conserving algorithm. The scheduling decisions, which define the sequence of execution of client tasks, are taken under the principal of EDF. In Fair-EDF, the

Algorithm 1: execution of task set using Fair-EDF on a single processor

Input: Task set including n periodic tasks and a single processor

Output: Task set schedule based on Fair-EDF

1. S is a server task with n clients tasks such that $U_S = \sum_{i=1}^n u_i$
//the maximum value of U_S is 1 i.e. $U_S \leq 1$
 2. $\sum_{l=0}^{t-1} X(S, l)$ is total executed time units by server task up to time t ,
 3. @ (each scheduling event)
 4. Sort the ready task list w. r. t earliest deadline first
 5. @ (each tick of time t)
 6. Compute $\text{lag}(S, t) = U_S * t - \sum_{l=0}^{t-1} X(S, l)$
 7. if ($|\text{lag}(S, t)| < 1$)
 8. Run the task with top priority
 9. else
 10. Preempt the running task
-

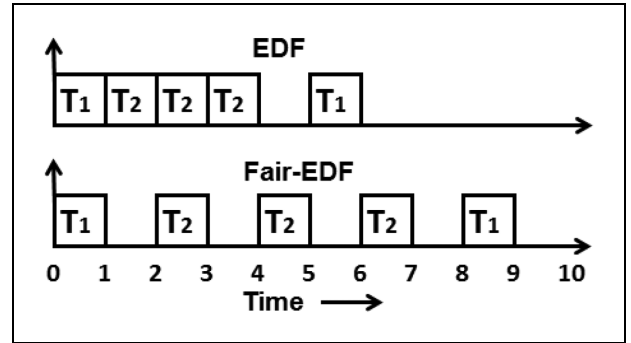


Figure 1. Schedule under EDF and Fair-EDF at 50% utilization factor on a single processor.

number of pre-emptions and context switches will less than the PFair²² because the constraint of the lag is applied to each task at every instant of time in PFair. However, in the case of Fair-EDF, it is applied only to the server that reduces the number of context switches. The pseudo code of Fair-EDF is defined in Algorithm 1. In classical EDF, the tasks are sorted with respect to earliest deadline at each scheduling event and the task with lowest deadline is assigned the processor. In Fair-EDF, the lag value for the server task is computed at each tick of time and thereafter the decision is taken whether to continue execution or to preempt it. This is mentioned at line 6 of the algorithm. The control on the execution pattern by adjusting the lag value results in uniform execution of time. Figure 1 shows the scheduling of a task set mention in Table 1 on a single processor. The value of U_S is 0.5. The workload is concentrated in EDF while Fair-EDF is uniformly distributed over the time. The uniform execution of the workload in Fair-EDF is achieved at the rate of higher computational overhead and the raised number of pre-

Table 1. Task set configuration.

Tasks	C_i	$d_i = P_i$
T_1	1	5
T_2	3	10

emptions than that of basic EDF scheduling. The cost of pre-emption is assumed to be zero in this work.

Properties of Fair-EDF

Theorem 1. Fair-EDF is optimal in all scheduling problems where the EDF is optimal

$$U = \sum_{i=1}^n u_i \leq 1 \quad (13)$$

The Fair-EDF performs the scheduling of a task set on a processor such that at any given time t , accumulated time allocated to a server with utilization factor U_S is either $t*U_S$ or $t*U_S$. It means that in Fair-EDF, a task may not be more than a unit time away from its ideal execution at any given time. If this condition is fulfilled, all the deadlines will be met and task set will be schedulable. It is demonstrated in the corollary 1 and 2.

Corollary 1. The server S will meet the deadline if it follows Fair-EDF.

When a server is running, actually one of its client tasks is running on the processor. Suppose, H is the hyper period of the task set in the server which can be regarded as the time period of the server. The sever S requires U_S*H , units of time at $t = H$. The lag function can be rewritten as equation (14)

$$lag(S, H) = E - \sum_{l=0}^{H-1} X(S, l) \quad (14)$$

where E is the total number of task set units to be completed in hyper period H . Both E and $\sum_{l=0}^{H-1} X(S, l)$ are integers. The only possibility to satisfy the lag condition given in equation is that both have the same value and their difference is zero. In other words, both the practical and fluid schedules converge on the same point. Hence, the server will meet its deadline.

Corollary 2. All the client tasks of server S will meet their individual deadlines by following Fair-EDF.

The individual utilization factor of any client task u_i cannot be more than the server utilization factor U_S , and EDF is used for their internal scheduling.

Table 2. H.264 video decoder application tasks.

Task	r_i	$d_i = P_i$	C_i
TG	0	15	02
SI	15	15	03
RE-1	30	30	17
RE-2	30	30	17
RE-F	60	30	08
LI	90	30	03
RA	120	30	02

Therefore, each client task will definitely meet its deadline if the server meets the deadline.

Results

A statistical approach has been used to evaluate the strength of Fair-EDF by testing it over different target applications in uni-processor as well as multi-processor environment. In the multi-processor environment, a partitioned algorithm based on Fair-EDF has been evaluated. The performance has been evaluated on the basis of efficiency of an algorithm to reduce temperature peaks, variation and averages for the same set of task sets. A parameter of the spatial gradient is added in case of multi-processor environment. The spatial gradient is defined as the temperature difference between the hottest and the coolest cores at any given time.

The simulation setup comprises a scheduling Simulation TOol for Real-time Multi-processor scheduling (STORM)²⁷ and a thermal modelling tool ATMI.²⁵ The duration of each simulation is 800 s. We used Roger Stafford's²⁸ randfixedsum algorithm to generate the task set of variable total utilization factor for the evaluation process. A MATLAB implementation of the algorithm is publicly available with all necessary documentation. We also tested the algorithm on for H.264 video decoder application with eight tasks. Table 2 gives the complete task set of H.264 and their parameters.

Uni-processor platform

Figure 2(a) and (b) shows a comparison of average thermal behaviour of EDF and Fair-EDF on synthetic task set at 45% and 60% total utilization factors U , respectively. The results show that Fair-EDF always gives equal or better performance than EDF algorithm in controlling the thermal issues. Fair-EDF reduces the temperature peaks up to 5% at different values of total utilization factor U . The Fair-EDF also always incurs less or equal temperature variations when compared with the EDF. This is due to the work-conserving behaviour of EDF which introduces large idle intervals after completion of assigned workload. The Fair-EDF gives

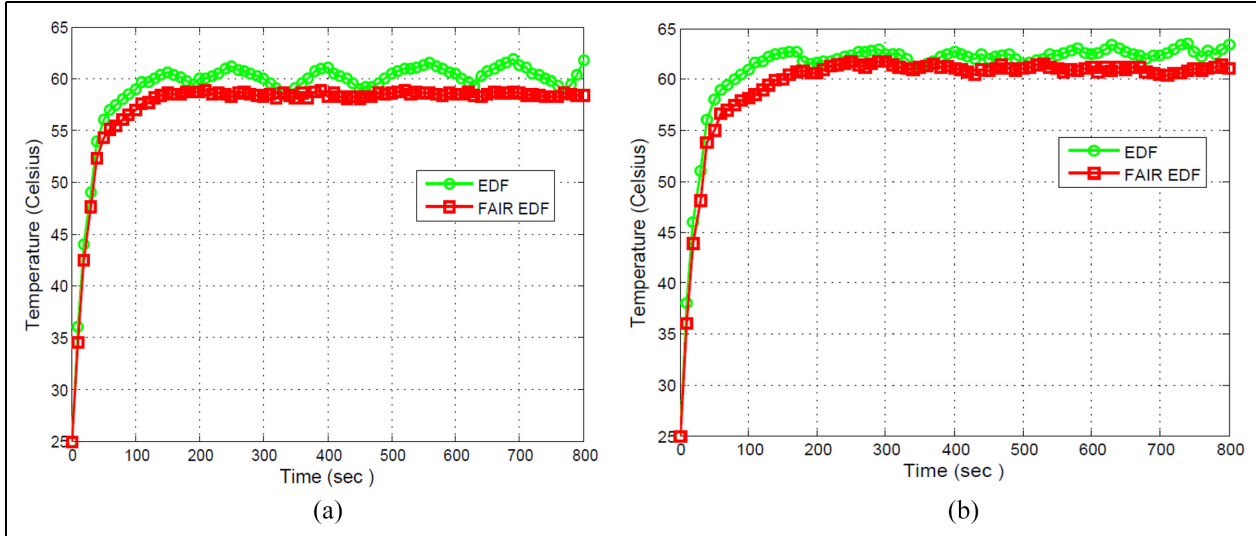


Figure 2. (a) Thermal profile of EDF and Fair-EDF at 45% workload and (b) thermal profile of EDF and Fair-EDF at 60% workload.

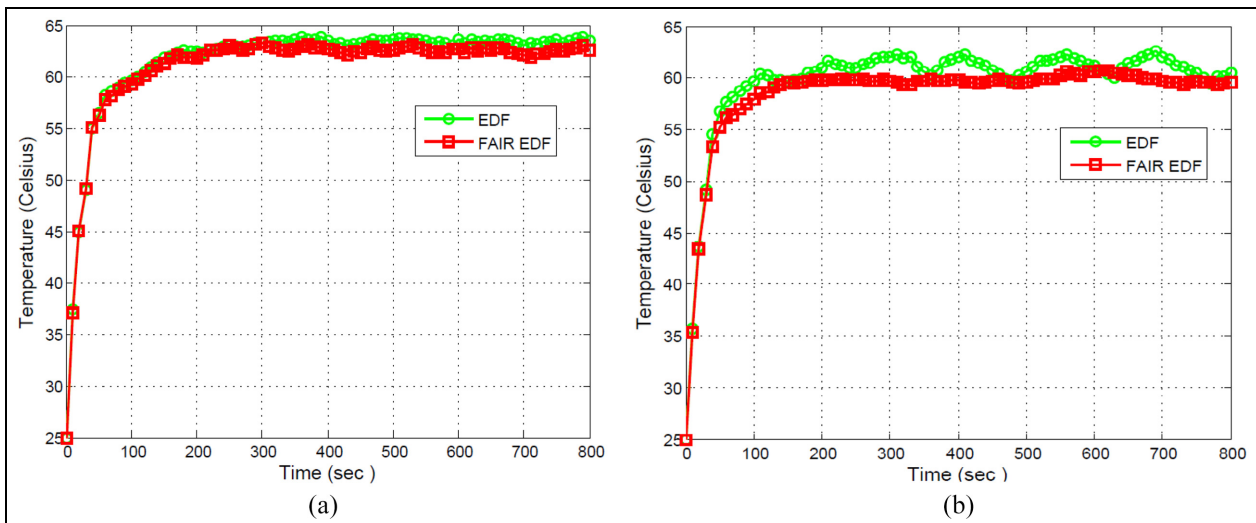


Figure 3. (a) Thermal profile of EDF and Fair-EDF at 70% workload and (b) thermal profile of EDF and Fair-EDF with video decoder application.

better performance when the processor is relatively less busy. With an increase in the value of total utilization factor U , Fair-EDF has less room for improvement. The performance of Fair-EDF is better at 45% in Figure 2(a) than at 60% in Figure 2(b). When the processor utilization becomes $U = 1$, the processor always remain busy and Fair-EDF works exactly the same as EDF algorithm.

Figure 3(a) shows the comparison between EDF and Fair-EDF at 75% total utilization factors, while Figure 3(b) shows the comparison between the two same algorithms for H.264 video decoder application with eight tasks at 45% total utilization factor. The results show that Fair-EDF outperforms EDF in both

cases. However, the performance of Fair-EDF is relatively better at lower utilization factor.

Multi-processor platform

Figures 4 and 5 show a comparison of average thermal behaviour of partitioned algorithm based on proposed Fair-EDF with thermal balancing policy algorithm,²⁹ predictive thermal management policy³⁰ and global EDF at 25% and 45% total utilization factors U over four cores system. We have compared the algorithms on the basis of temperature temporal gradient, temperature spatial gradient and average core temperature.

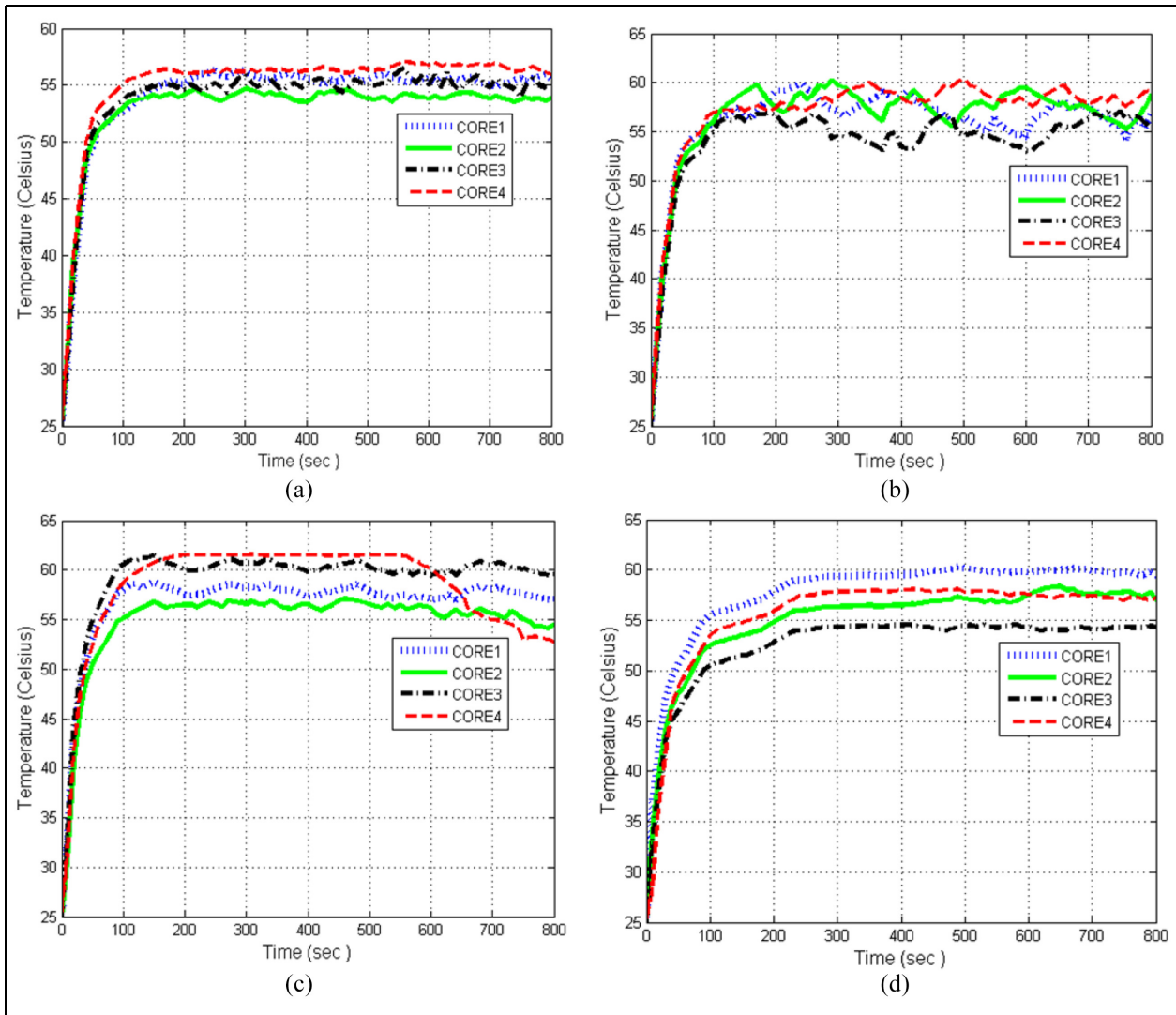


Figure 4. (a) Thermal profile of partitioned algorithm (Fair-EDF) at 25% workload on four cores multi-processor system, (b) thermal profile of global EDF at 25% workload on four cores multi-processor system, (c) thermal profile of PDTM at 25% workload on four cores multi-processor system and (d) thermal profile of TBP at 25% workload on four cores multi-processor system.

Temperature temporal gradient. Temperature temporal gradient is the variation of temperature on any core with respect to time. It negatively affects the performance of the system. The thermal variation on any single core of the multi-core system in case of partitioned algorithm is comparable with thermal balancing policy. Moreover, partitioned algorithm incurs significantly less temporal gradient than both global EDF algorithm and predictive thermal management policy. This trend is valid both for 25% and 45% total utilization factors. However, the lesser the workload the higher the performance gap between the partitioned algorithm and other algorithms. The temporal gradient/100 s for partitioned algorithm based on Fair-EDF is 1.5°C and 2°C at 45% and 25% total utilization factor, respectively, which is better than the performance of other algorithms in this regard. These results are summarized in Table 3.

Temperature spatial gradient. Spatial gradient is the temperature difference between the hottest and the cooling cores of multi-core systems at any particular moment. The results in Figures 4 and 5 show that partitioned algorithm based on the proposed Fair-EDF outperforms all the other algorithms in this regard. The reason being that Fair-EDF ensures stable thermal problem of each core; hence, lowers the value of spatial gradient. This trend is valid both for 25% and 45% total utilization factors. Table 4 gives the maximum value of spatial gradient at any given time. The spatial gradient for partitioned algorithm is 3.5°C and 2°C for 45% and 25% total utilization factors, respectively, which is better than the performance of other algorithms in this regard.

Average core temperature. Table 5 gives the average temperature of all the four cores in multi-processor system

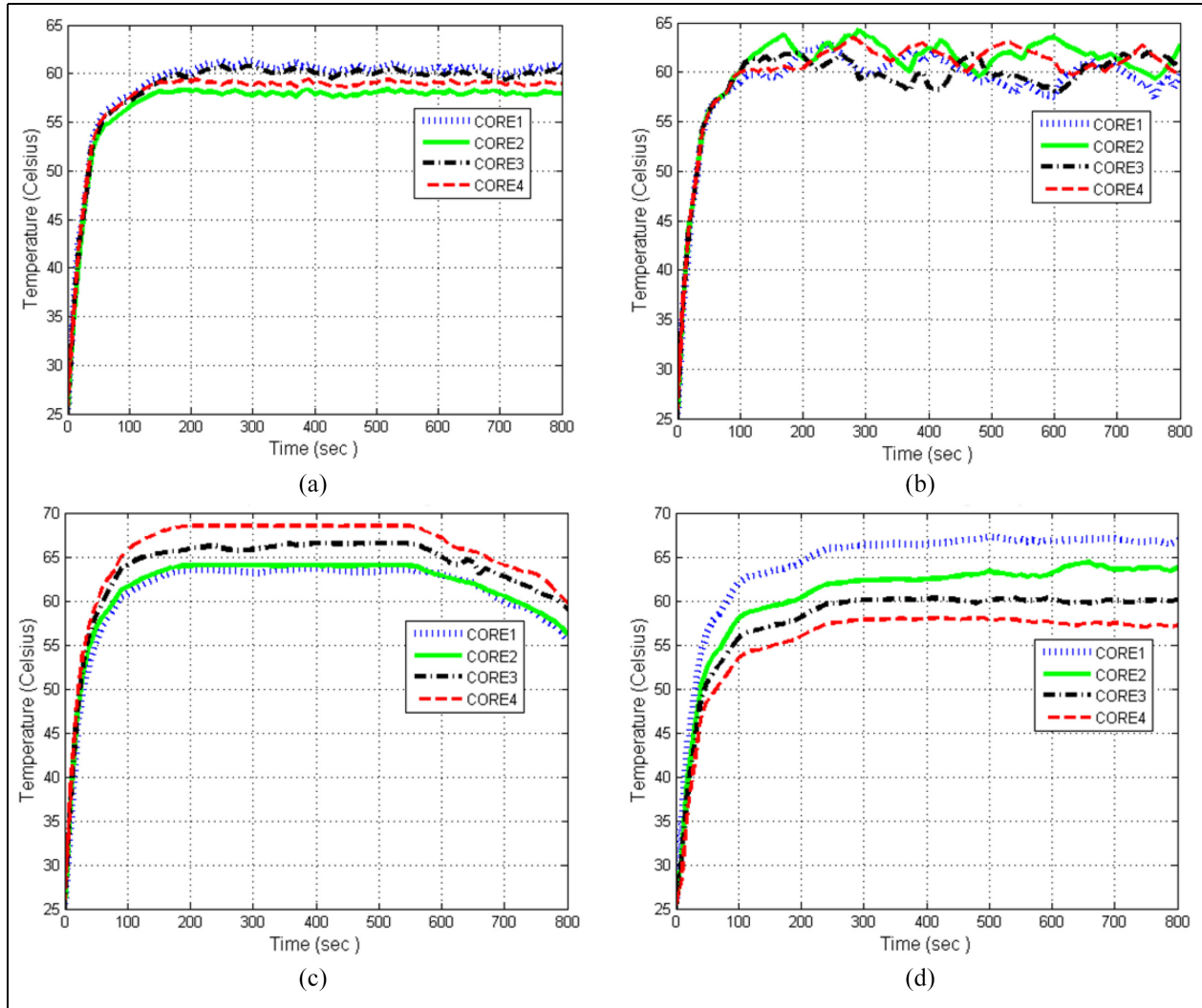


Figure 5. (a) Thermal profile of partitioned algorithm (Fair-EDF) at 45% workload on four cores multi-processor system, (b) thermal profile of global EDF at 45% workload on four cores multi-processor system, (c) thermal profile of PDTM at 45% workload on four cores multi-processor system and (d) thermal profile of TBP at 45% workload on four cores multi-processor system.

Table 3. Maximum value of temporal gradient/100 s interval in any of the four cores.

Algorithm	45% workload	25% workload
TBP	2°C	2°C
Global EDF	4°C	5°C
PDTM	4°C	4°C
Partitioned algorithm (Fair-EDF)	1.5°C	2°C

TBP: thermal balancing policy; PDTM: predictive thermal management policy; EDF: earliest deadline first.

after 100 s. The partitioned algorithm based on the proposed Fair-EDF algorithm gives equal or lower average temperature than other algorithms. The partitioned algorithm gives an average temperature of 56°C when

total utilization factor is 25% and 60°C when total utilization factor is 45%.

Conclusion

This article presents Fair-EDF, a temperature-aware scheduling algorithm for uni-processor platform. It improves the thermal stability of EDF algorithm by adding the fairness for smooth execution of workload without violating the optimality of the classical EDF algorithm. The proposed algorithm is evaluated for temperature peaks, variations and the average temperature in uni-processor system and is assessed for temperature temporal gradients, temperature spatial gradients and average temperature in a multi-processor environment. The results verify the efficiency of the proposed algorithm. It reduces the temperature

Table 4. Maximum value spatial gradient in four core multi-processor system.

Algorithm	45% workload	25% workload
TBP	9°C	5°C
Global EDF	7°C	5°C
PDTM	3°C	5°C
Partitioned algorithm (Fair-EDF)	3.5°C	2°C

TBP: thermal balancing policy; PDTM: predictive thermal management policy; EDF: earliest deadline first.

Table 5. Average temperature of four cores in multi-processor system.

Algorithm	45% workload	25% workload
TBP	61°C	56°C
Global EDF	60°C	57°C
PDTM	66°C	57°C
Partitioned Algo (Fair-EDF)	60°C	56°C

TBP: thermal balancing policy; PDTM: predictive thermal management policy; EDF: earliest deadline first.

variations, average temperature and temperature peaks up to 5% in uni-processor environment. In the case of multi-processor environment, it performs better than the predictive thermal management policy, thermal balancing policy and the global EDF. It gives maximum 2°C per 100 s temporal gradient, and maximum 3.5°C of spatial gradient. The algorithm is more effective at lower values of utilization factor and become lesser efficient at higher values of utilization factor. Moreover, the technique is pure software-based solution and does not require any specific hardware which makes it suitable for use in diverse environment of IoTs.


Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) received no financial support for the research, authorship, and/or publication of this article.

ORCID iD

Adeel Anjum  <https://orcid.org/0000-0001-5083-0019>

References

1. Gunther S, Binns F, Carmean DM, et al. Managing the impact of increasing microprocessor power consumption. *Intel Tech J* 2001; 5(1), 1–9.
2. Rosing TS, Mihic K and De Micheli G. Power and reliability management of SoCs. *IEEE Trans VLSI Syst* 2007; 15(4): 391–403.
3. Srinivasan J and Adve SV. Predictive dynamic thermal management for multimedia applications. In: *Proceedings of the 17th annual international conference on supercomputing (ICS '03)*, San Francisco, CA, 23–26 June 2003, pp.109–120. New York: ACM Press.
4. Burd T, Pering T, Stratakos A, et al. A dynamic voltage scaled microprocessor system. *IEEE J Solid State Cir* 2000; 35: 1571–1580.
5. Lee JS, Skadron K and Chung SW. Predictive temperature-aware DVFS. *IEEE Trans Comput* 2010; 59(1): 127–133.
6. Baati K and Auguin M. Temperature-aware DVFS-DPM for real-time applications under variable ambient temperature. In: *8th IEEE international symposium on industrial embedded systems*, Porto, 9–21 June 2013, pp.13–20. New York: IEEE.
7. Mahmood A, Khan SA, Albalooshi F, et al. Energy-aware real-time task scheduling in multiprocessor systems using a hybrid genetic algorithm. *Electronics* 2017; 6: 40.
8. Nogues E, Pelcat M, Menard D, et al. Energy efficient scheduling of real time signal processing applications through combined DVFS and DPM. In: *24th Euromicro international conference on parallel, distributed, and network-based processing (PDP)*, 2016, Heraklion, 17–19 February 2016, pp.622–626. New York: IEEE.
9. Zanini F, Aienza D and De Michele G. A control theory approach for thermal balancing of mp soc. In: *Proceedings of the 2009 Asia and South Pacific design automation conference (ASP-DAC)*, Yokohama, Japan, 19–22 January 2009, pp.37–42. New York: IEEE.
10. Kursun E and Cher C-Y. Variation-aware thermal characterization and management of multi-core architectures. In: *International conference on computer design (ICCD)*, Lake Tahoe, CA, 12–15 October 2008, pp.280–285. New York: IEEE Computer Society.
11. Salamy H. Task scheduling on multicore embedded systems under power and thermal constraints. *Int J Electronic* 2015; 102(12): 2075–2091.
12. Merkel A, Bellosa F and Weissel A. Event-driven thermal management in SMP systems. In: *Second workshop on temperature-aware computer systems (TACS'05)*, Madison, WI, 4–8 June 2005. CiteSeer.
13. Bashir Q, Shehzad MN, Awais MN, et al. A scheduling based energy-aware core switching technique to avoid thermal threshold values in multi-core processing systems. *Microprocess Microsyst* 2018; 61: 296–305.
14. Bashir Q, Shehzad MN, Awais MN, et al. An online temperature-aware scheduling technique to avoid thermal emergencies in multiprocessor systems. *Comput Electric Eng* 2018; 61: 83–98.
15. Zhou J, Wei T, Chen M, et al. Thermal-aware task scheduling for energy minimization in heterogeneous real-time MPSoC systems. *IEEE Trans Comput-Aided Des Int Cir Syst* 2016; 35(8), 1269–1282.
16. Alsubaihi S and Gaudiot J-L. PETRAS: performance, energy and thermal aware resource allocation and scheduling for heterogeneous systems. In: *Proceedings of the*

- 8th international workshop on programming models and applications for multicores and manycores (PMAM'17), Austin, TX, 4–8 February 2017, pp.29–38. New York: ACM Press.
17. Alsafrjalani MH and Adegbija T. TaSaT: thermal-aware scheduling and tuning algorithm for heterogeneous and configurable embedded systems. In: *Great Lakes symposium on VLSI (GLSVLSI '18)*, Chicago, IL, 23–25 May 2018. New York: ACM Press.
 18. Guo Z, Bhuiyan A, Saifullah A, et al. Energy-efficient multi-core scheduling for real-time DAG tasks. In: *LIPICs-Leibniz international proceedings in informatics (76 Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik)*, Dubrovnik, 30 June 2017. Wadern: Dagstuhl Publishing.
 19. Bhuiyan A, Guo Z, Saifullah A, et al. Energy-efficient real-time scheduling of DAG tasks. *ACM Trans Embed Comput Syst* 2018; 17: 84.
 20. Iranfar A, Kamal M, Afzali-Kusha A, et al. TheSPoT: thermal stress-aware power and temperature management for multiprocessor systems-on-chip. *IEEE Trans Comput-Aided Des Int Cir Syst* 2017; 37: 1532–1545.
 21. Ahmed R, Ramanathan P and Saluja K. On thermal utilization of periodic task sets in uni-processor systems. In: *19th international conference on embedded and real-time computing systems and applications*, Taipei, Taiwan, 19–21 August 2013. New York: IEEE.
 22. Ahmed R, Ramanathan P and Saluja K. Necessary and sufficient conditions for thermal schedulability of periodic real-time tasks under fluid scheduling model. *ACM Trans Embed Comput Syst* 2016; 15: 49.
 23. Baruah SK, Cohen NK, Plaxton CG, et al. Proportionate progress: a notion of fairness in resource allocation. *Algorithmica* 1996; 15: 600–625.
 24. Jejurikar R, Pereira C and Gupta R. Leakage aware dynamic voltage scaling for real-time embedded systems. In: *Proceedings of the 41st annual design automation conference*, San Diego, CA, 7–11 July 2004, pp.275–280. New York: ACM Press.
 25. Michaud P and Sazeides Y. ATMI: analytical model of temperature in microprocessors. In: *Proceedings of third annual workshop on modeling, benchmarking and simulation (MoBS '07)*, San Diego, CA, 10 June 2007, pp.12–21. CiteSeer.
 26. Regnier P, Lima G, Massa E, et al. Run: optimal multiprocessor real-time scheduling via reduction to uniprocessor. In: *Proceedings of the 32nd IEEE real-time systems symposium*, Vienna, 29 November–2 December 2011, pp.104–115. New York: IEEE.
 27. Urunuela R, Deplanche A-M and Trinquet Y. STORM – a simulation tool for real-time multiprocessor scheduling evaluation. In: *Proceedings of IEEE international conference on emerging technology and factory automation (ETFA)*, Bilbao, 13–16 September 2010, pp.1–8. New York: IEEE.
 28. Stafford R. Random vectors with fixed sum, 2006, <http://www.mathworks.com/matlabcentral/fileexchange/9700>
 29. Yang J, Zhou X, Chrobak M, et al. Dynamic thermal management through task scheduling. In: *International symposium on performance analysis of systems and software (ISPASS)*, Austin, TX, 20–22 April 2008. New York: IEEE.
 30. Yeo I, Liu CC and Kim EJ. Predictive dynamic thermal management for multicore systems. In: *Proceedings of the 45th annual design automation conference (DAC '08)*. Anaheim, CA, 8–13 June 2008, pp.734–739. New York: ACM Press.