International Journal of Reconfigurable and Embedded Systems (IJRES)
Vol. 7, No. 1, March 2018, pp. 48~56
ISSN: 2089-4864, DOI: 10.11591/ijres.v7.i1.pp48-56

Thermal Analysis of Fair Scheduling in Real-time Embedded Systems

Tayyaba Bokhari[1], Sajjad Haider Shami[2], Farhan Haseeb[3]
[1]Department of Electrical Engineering, University of Lahore, Lahore, Pakistan
[2]Departement of Electrical Engineering, University of Management and Technology, Lahore, Pakistan
[3]Exponento Private Limited, Lahore, Pakistan

Article Info

Article history:

Received Dec 7, 2017
Revised Feb 5, 2018
Accepted Feb 21, 2018

Keywords:

DP-Fair
Embedded systems
Fair scheduling
PFair
Real-time

ABSTRACT

Over the past few decades, increased demand of highly sophisticated real-time applications with complex functionalities has directly led to exponentially increased power consumption and significantly elevated system temperatures. These elevated temperature and thermal variations present formidable challenges towards system reliability, performance, cooling cost and leakages. This article explores the thermal management strength of two fairness based algorithms, namely Proportional Fair (PFair) and Deadline Partitioning Fair (DP-Fair). In related literature, the introduction of fairness is often considered as a tool to achieve optimality in multiprocessor scheduling algorithms. This work shows that these algorithms bring about better thermal profile when compared with the commonly used Earliest Deadline First (EDF) algorithm in similar conditions both in uniprocessor and multiprocessor environments. A simulation is conducted for periodic task set model. The obtained results are encouraging and show that use of fairness based algorithms reduces the operating temperature, peak temperature, and thermal variations.

Corresponding Author:

Tayyaba Bokhari,
Departement of Electrical Engineering,
The University of Lahore,
1-Km Defence Road, Lahore, Pakistan.
Email: tayyaba.bukhari@ee.uol.edu.pk

1. INTRODUCTION

The recent advancements in technology have enabled integration of more and more computing resources on a single chip to accommodate the increasing performance demands. More than 40 billion transistors are being integrated into a single 300 mm^2 die [1]. This increased chip density and growing complexity of real-time applications pose a challenge, i.e., exponentially increasing power consumption rate of computing devices. Moreover, due to this high power consumption and continuous scaling down in geometries, the available chip surface for heat dissipation is decreasing which is resulting in increased power densities and large temperature variations on chip [2]. This increased power usage causes two types of difficulties: higher energy consumption and increased device temperature. A large part of energy in these circuits is converted into heat and eventually increases the device temperature.

Chip temperature is a function of chip power density. High temperatures reduce the effective operating speed of devices. Operating speed of the device decreases due to increase in local resistance resulting from high temperature. This eventually affects the overall performance. In CMOS circuits, clock timing is very sensitive to temperature. A 15^oC increase in temperature can add 10% to 15% delay in the circuit [3]. The reliability, performance, and cooling cost of the system is also affected by temperature. Processor failure rate is also closely related with temperature. A rise in the operating temperature of $10-15^oC$ can result in a 2x difference in the life span of the devices [4]. Leakage is another factor that is related to

temperature. A positive feedback loop exists between leakage and temperature. In 65nm technology, if a chip temperature rises from 60^{o}C to 80^{o}C, leakage power increases by 21% [5]. This can damage the circuit if not properly handled. These issues in System-on-Chip (SoC) arise due to two major thermal phenomena [6]:

- Creation of thermal cycles caused by temperature variations between upper and lower bounds.
- Creation of hotspots caused by concentrated on-chip computation activity at any particular core/component level.

Thermal cycles are created due to workload variation, low frequency power changes or due to some power management assessment like putting the system into deep sleep mode in idle time units. This phenomenon of thermal cycles has correlation with the magnitude and frequency of the temperature cycles on the core. These cycles cause accelerated package fatigue and plastic deformation of materials that accumulate at each temperature cycle and can lead to cracks and other permanent failures. Similarly, in multiprocessor systems, a certain processing element may remain functional for longer duration compared to others; such an element carries out extra computational activity and converts more energy into heat for itself and its neighbouring components. This leads to the creation of hotspots which exponentially speed up many aging mechanisms such as electro-migration, stress migration, and dielectric breakdowns, ultimately causing permanent device failures [4].

Many temperature management techniques have been proposed in real-time embedded systems to deal with thermal issues both at hardware and software levels, but software based solutions are advantageous in terms of cost. These software based techniques perform the scheduling decisions based on theoretical knowledge by playing with different system parameters without missing the deadlines. The motivation for conducting this study is to analyse a specific software based mechanism for thermal management. This technique uses fairness or fair scheduling as a tool to counter the thermal issues in embedded systems. The fairness based algorithms distribute the workload over time instead of concentrating the same at some point.

This type of execution of workload over time lowers the operating temperature, peak temperature, and thermal variations at the same time. To the best of the authors' knowledge, there is no published work that uses fairness based techniques to address thermal issues in real-time embedded systems. While the main goal was to study the effect of fairness on the thermal behaviour of the system, two specific fairness based algorithms were selected for investigation i.e., PFair [7] and DP-Fair [8].

The rest of the paper is organized as follows. In Section 2 related work is briefly discussed. An overview of fairness and fairness based algorithms are presented in Section 3. Section 4 describes the simulation model and results are presented and discussed in Section 5. Section 6 concludes the article.

2. RELATED WORK

Thermal efficiency in real-time systems is a well-known problem. Hardware, software and hybrid solutions have been proposed in literature to address these thermal issues. Some hardware based techniques are being commercially used to resolve such thermal issues. Intel's Pentium 4 uses a cooling fan and heat spreader to lower the chip temperature [9]. A number of techniques to control the peak temperature and thermal gradients through hardware and software cooperation are presented in [10, 11]. These techniques acquire the temperature values using physical sensors and perform scheduling decisions accordingly.

Some purely scheduling-based solutions to avoid thermal hotspots and thermal variations are presented in [5, 12]. These techniques optimize scheduling by playing with different system parameters without missing the target deadlines.

For real-time embedded multiprocessor systems, Mulas et al. [13] proposed a task migration technique to balance the temperature. This technique uses the coolest core as destination core to achieve much more balanced temperature as compared to energy balancing techniques presented in [14]. Coskun et. al. [10] proposed a scheduling technique that uses Integer Linear Programming (ILP) for real-time Multi-Processor Systems-on-Chip (MPSoCs) that significantly minimizes thermal hotspots and spatial gradients as compared to only heat minimization technique. In this technique, tasks are not assigned to the adjacent cores at the same time and this also minimizes the duration of thermal emergencies. In [11], this technique is combined with a coolest Function-Level Processor (FLP) technique, called hybrid technique. Another temperature aware task scheduling technique for hard real-time applications in MPSoCs is proposed by Chantem et al. [15] that apply Mixed-Integer Linear Programming (MILP) to schedule and assign hard real-time tasks. This technique considers both temporal and spatial temperature variations and reduces the temperature by 8.75 ºC on average. However, this technique has a limitation that it is not scalable with the growing size of the problem. Zhao et al. [16] proposed a thread migration technique to reduce peak

temperature and thermal variance in 3D multi-core processor and stacked DRAM. This technique reduces peak temperature of processor, on average, by 4.7 ºC and thermal variaion by 4.48 ºC.

Certain Dynamic Power Management (DPM) and Dynamic Voltage and Frequency Scaling (DVFS) techniques to control the temperature have been proposed by various researchers. A DVFS based temperature control method that predicts the local temperature of chip and performs the voltage or frequency scaling accordingly was proposed by Lee et al. [17]. Baati and Michel [18] proposed a DVFS-DPM based temperature aware technique to address the thermal problems, using some heuristics. This study explores the ability of fairness based techniques to alleviate thermal issues in real-time embedded systems.

3. INTRODUCTION TO FAIRNESS

The concept of fairness is closely related to that of fluid schedule where a task completes its execution time with uniform rate in its deadline. A task T_i at any time t executes exactly for $u_i * t$ time units, where u_i is utilization factor of task T_i. However, 100% fluid scheduling is not possible because a processor runs either at full capacity or zero. Moreover, a processor cannot execute multiple tasks at any given time. Hence, fairness is used to achieve maximum possible behaviour of the fluid schedule. Fairness is practically achieved by controlling the task execution and restricting its deviation from the fluid (ideal) behaviour. The deviation or difference between ideal and practical execution of each task is measured with the help of a lag function. The lag for a task T_i at time t is defined as:

$$lag\ (T_i\ , t) = u_i * t - \sum_{l=0}^{t-1} S(T_i, l) \qquad (1)$$

Where $u_i * t$ represents the total executed time of task at time t in an ideal system. S is the real schedule function, $S(T_i, l)$ = 1 means that task is scheduled in the slot l while, $S(T_i, l)$= 0 shows that task is not scheduled over the slot.

The concept of fairness is illustrated in Figure 1 where a task, with utilization factor 0.5 and deadline 10, is scheduled on a single processor first with the help of EDF (Earliest Deadline First) scheduling and then with fairness based scheduling algorithm. When EDF is used, the task is completed in the first five slots of the scheduling duration. On other hand, the fairness based algorithm schedules the task proportional to its utilization factor or close to the fluid schedule. For example, when t = 8, it executes 4 units which it would have consumed in case of fluid schedule. However, fairness is achieved at the cost of some extra preemptions as shown in Figure 1.

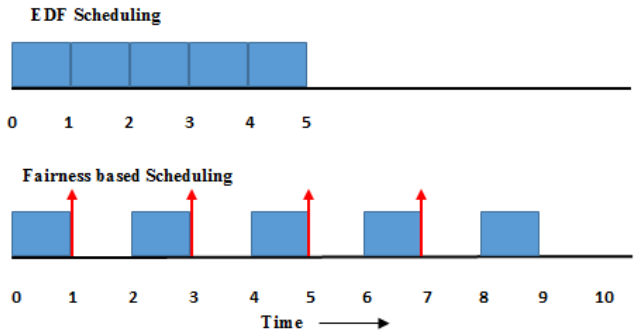

Figure 1. EDF and Fairness based scheduling

Different algorithms based on fairness use different value of lag. For example, PFair and DP-Fair, two fairness based optimal algorithms for multiprocessor systems, require different amount of fairness by asking for different value of lag function at different time. These two were chosen for comparison with EDF in this investigation, and are briefly described below.

Proportional-Fair (PFair) was proposed by Baruah et al. [7] in 1996. It restricts the value of lag function, defined in (1), to be less than or equal to one for all value of time and for each task. PFair performs the scheduling of tasks on the processors such that at any given time t, the accumulated time allocated to a task T_i with utilization factor u_i is either $\lceil u_i * t \rceil$ or $\lfloor u_i * t \rfloor$. In other words, in PFair a task may not be more than a quantum length away from its ideal execution in the fluid system at any time.

Deadline Partitioned Fair (DP-Fair) [8] combines the notion of fluid scheduling with the one of deadline partitioning to still guarantee optimality while lowering the preemptions than PFair. It relaxes the requirement of lag while still ensuring the optimality. In DP-Fair, the value of lag should be less than or equal to 1 only at the time of arrival of tasks in the system.

4. SIMULATION MODEL

To evaluate the potential of scheduling algorithms, a simulation model was designed within which a statistical approach was used, by testing them over a significant number of task sets. The simulation model is shown in Figure 2.

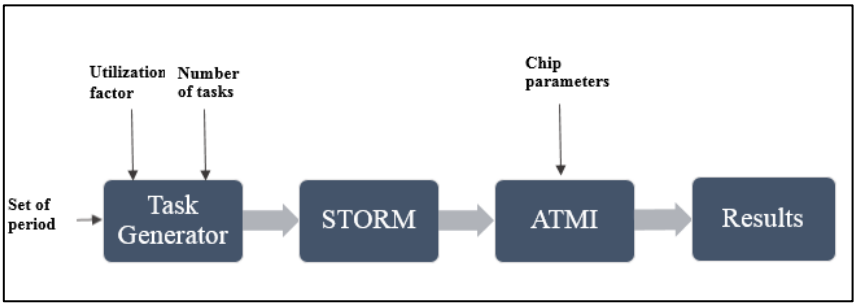

Figure 2. Simulation Model

The model consists of a task generator, a scheduling simulator Simulation TOol for Real-time Multi-processor scheduling (STORM) [19] and a temperature modelling simulator Analytical model of Temperature in Microprocessors (ATMI) [20]. The task sets are produced randomly using the task set generator under different constraints, set by the user. Roger Stanford's randfixedsum [21] is the core of the task generator that generates a fixed number of random real values within an interval [a, b]. The task set generator takes three inputs: total utilization factor U, number of tasks in each task set N and set of periods Sp. At output it gives N couples of (C_i , P_i) and Least Common Multiple (LCM) of Sp i.e. hyper period (HP), where C_i is worst-case execution time of a task and P_i denotes its period which is equal to the deadline D_i. Each task has a utilization factor of $u_i = C_i / D_i$.

Task sets are given to the simulation tool STORM in the form of XML files. An XML file gives all the information about task set parameters, scheduling algorithm and hardware parameters. STORM simulates the XML file over the specific scheduling policy and gives power statistics of the core at each scheduling event, that are stored in text files.

The power profile in the form of text files are used with chip parameters in the thermal tool ATMI to generate estimated thermal profile of each core in the architecture. All thermal profiles generated by ATMI are relative to ambient temperature. ATMI was used for a two layer model of the chip MPC5517E (from Freescale Inc). The ATMI model of physical architecture is given in Figure 3. ATMI calculates the dynamic progression of the temperature at layer 1. The chip layout used and its thermal characteristics are those that are presented in [18]. These chip parameters are given in Table 1.

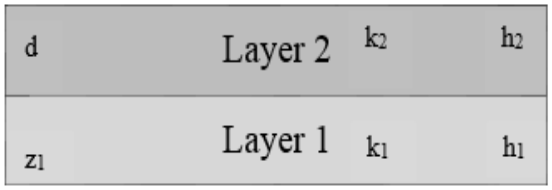

Figure 3. Model of chip under ATMI [22]

Table 1. Chip parameters for ATMI [17]

Parameter	Variable	Value
Layer 1 thickness	z_1	3e-4 (m)
Layer 2 thickness	$d = z_2 - z_1$	4e-4 (m)
Layer 1 thermal conductivity	k_1	148 (W/mK)
Layer 2 thermal conductivity	k_2	360 (W/mK)
Layer 1 thermal diffusivity	α_1	87e-6 (m^2/s)
Layer 2 thermal diffusivity	α_2	106.56e-6 (m^2/s)
Conductance b/w layer 1 and layer 2	h_1	500 (W/m^2K)
Conductance between layer 2 and ambient	h_2	50 (W/m^2K)
Width	L	6e-3 (m)

Both STORM and ATMI were used such that STORM reads power states of cores at each scheduling event defined by STORM while ATMI provides an estimate of temperature for each core in the architecture. Consequently, this co-simulation environment gave both the dynamic activity at each core (STORM) and the dynamic evolution of core's temperature (ATMI).

5. RESULTS

The results of simulation process are presented to ascertain the efficiency of fairness based techniques. The simulations were performed using a large data set with variable total utilization factor U, while considering architectures comprising of single core, 2-cores and 4-cores. Fairness based PFair and DP-Fair are first compared with EDF in single processor. Although PFair and DP-Fair are designed for multiprocessor platforms, these algorithms were also tested for single processor, because the contesting EDF is known for optimal performance for single processor environmement and a comparision in the context of temperature was well deserved.. In the plots shown ahead, the x-axis represents time units in milliseconds and y-axis represents the temperature in degree centigrade. The thermal profile gets stable after around initial 200 ms and reflects the behaviour of scheduling algorithms. The results show that fairness based algorithms generally show less variations, lower average temperature and fewer temperature peaks at different values of utilization factor.

a) **Single Processor:** In case of single processor, the thermal profiles are shown in Figure 4 and Figure 5 where the processor is occupied with 30% and 80% workload respectively. The plots show that DP-Fair gives around 2 oC lower temperature than EDF while PFair gives a relatively stabilized thermal behaviour than EDF and gives a stable profile.

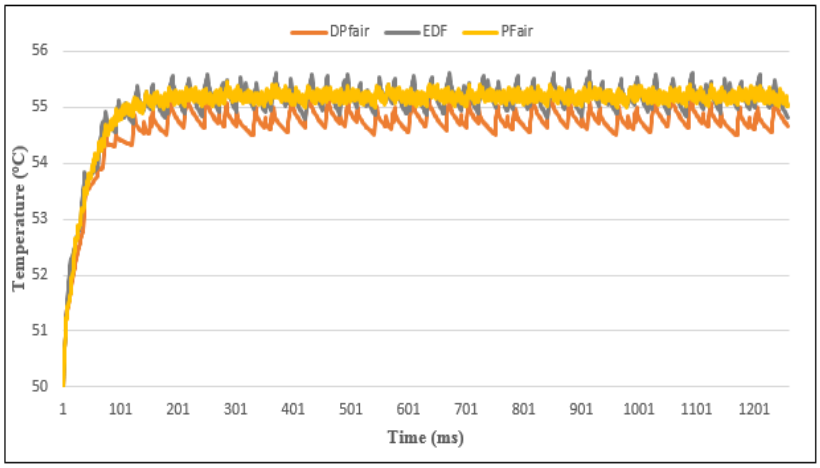

Figure 4. Single processor: Thermal profile at 30% workload

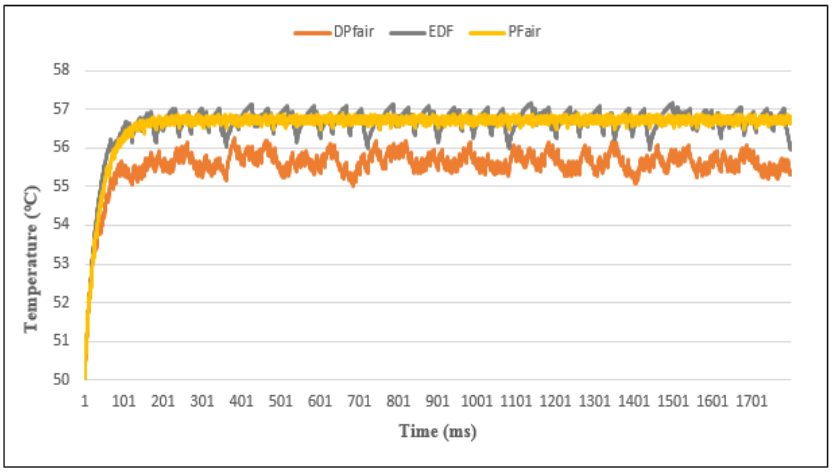

Figure 5. Single processor: Thermal profile at 80% workload

b) **Multiprocessor (2-core):** Figure 6 and Figure 7 show the thermal profile of 2-core processor at 30% and 70% workload respectively.

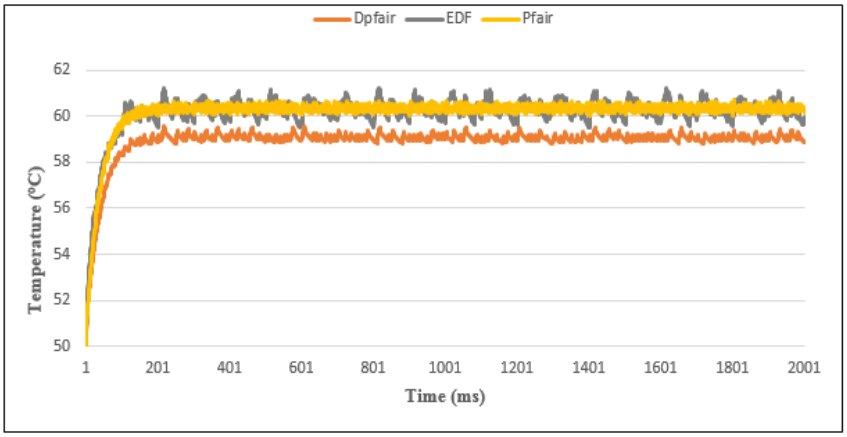

Figure 6. Two-core processor: Thermal profile at 30% workload

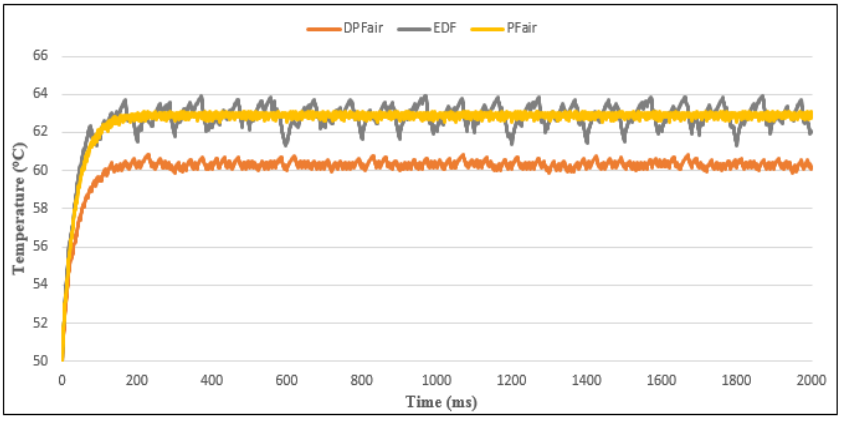

Figure 7. Two-core processor: Thermal profile at 70% workload

c) Multiprocessor 4-core: Figure 8 and Figure 9 show the thermal profile of 4-core processor at 30% and 60% workload respectively.

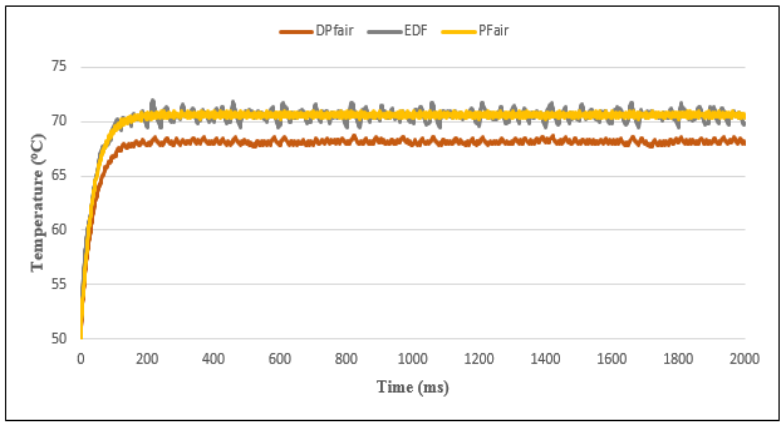

Figure 8. Four-core processor: Thermal profile of at 30% workload

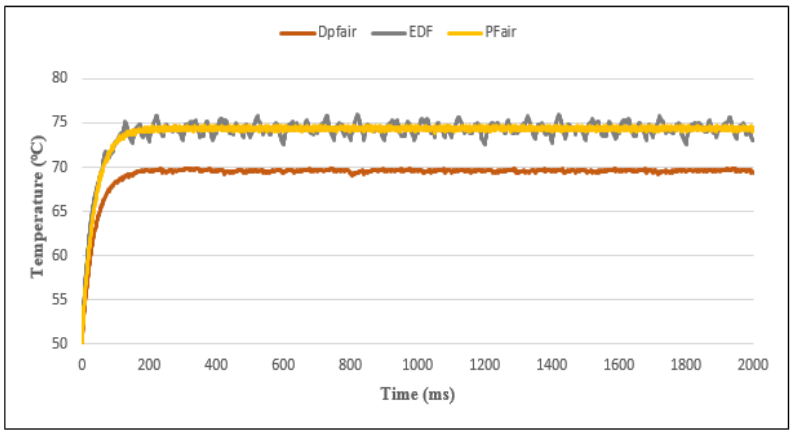

Figure 9. Four-core processor: Thermal profile at 60% workload

In case of multiprocessor environment, the thermal profiles were obtained with variable percentages of workload. The plots show that once again DP-Fair gives lower average temperatures hence avoid temperature peaks. It gives around 2 ^{o}C-5 ^{o}C lower temperature than global EDF.Although, PFair gives higher value of average temperature than DP-Fair but it shows a stable thermal profile as compare to the global EDF.

Although, PFair and DP-Fair are optimal algorithms for multiprocessor cases but in this work these algorithms have been compared with global EDF only at the utilization factor values where the global EDF provides feasible solutions i.e. (m+1)/2 where m is the number of processors. Here the term global refers global scheduling for multiprocessor, in which single system-wide queue of ready tasks is maintained and tasks are extracted at run-time to be scheduled at available processor. The average core temperature increases with the increase in the number of cores at the same percentage workload. At 30% workload, the core temperature in case of single processor is around 55 ^{o}C which rises to 70 ^{o}C in case of four-processor system.

6. CONCLUSION

This research investigates scheduling solutions to thermal issues in real-time embedded systems that occur mainly because of high power densities in a microprocessor system. For background, software based solutions at real-time operating system level are discussed with their ability to resolve thermal issues at

design time. Fairness based algorithms were analysed as a tool to counter these issues. In literature, fairness is predominantly used as a tool to achieve optimality in multiprocessor algorithms but this work has highlighted its properties in controlling thermal problems. A simulation based comparison of two fairness based algorithms (PFair and DP-Fair) was made with the most commonly used EDF algorithm on uniprocessor as well as multiprocessor systems under varying utilization factor. The results show that fairness based algorithms show a significant reduction in average operating temperature, peak temperature and thermal variations on the core in both cases. DP-Fair reduces the average temperature by 2 ^{o}C-5 ^{o}C and enables less temperature variations. PFair shows the best behaviour against thermal variations among these algorithms. In addition, the pattern of the results is generally consistent for different sets of task periods and utilization factors.

REFERENCES

[1] S. Borkar, "Thousand core chips: A technology perspective," in Proc. 44th Annual Design Automation Conf. (DAC07), ACM Press, New York, pp. 746-749, 2007.

[2] R. Viswanath, V. Wakharkar, A. Watwe, and V. Lebonheur, "Thermal Performance Challenges from Silicon to Systems," Intel Technology Journal, Q3, 2000.

[3] M. Santarini, "Thermal integrity: A must for low-power IC digital design." EDN, pp. 37-42, September 2005.

[4] A.K. Coskun, T. S Rosing, K. Mihic, Y. Leblebici and G. De Micheli. "Analysis and optimization of MPSoC reliability," Journal of Low Power Electronics (JOLPE), vol.2 no.1, pp.56-69, April 2006.

[5] W. Liao, L. He and K. Lepak , "Temperature and supply voltage aware performance and power modelling at micro architecture level," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 24 , no.7, pp. 1042 – 1053, 2005.

[6] "Failure Mechanisms and Models for Semiconductor Devices," JEDEC Publication JEP122C.

[7] N.C.G. Plaxton, S. Baruah and D.Varvel, "Proportionate progress: a notion of fairness in resource allocation," Algorithmica, 15, pp. 600-625, 1996.

[8] G. Levin, Shelby Funk, Caitlin Sadowski, Ian Pye, and Scott Brandt, "DP-fair: A simple model for understanding optimal multiprocessor scheduling," in Proc. 22nd Euromicro Conf. on Real-Time Systems, pp. 3–13, 2010.

[9] HP Corporation, Intel Corporation, Microsoft Corporation, Phoenix Tech., Ltd., and Toshiba Corporation., Advanced Configuration and Power Interface Specification, [Online]. Available http://www.acpi.info

[10] A. K. Coskun, T. S. Rosing and K. Whisnant, "Temperature aware task scheduling in MPSoCs," in Proc. Design, Automation and Test in Europe Conference and Exhibition (DATE '07), pp. 1-6, 2007.

[11] A. K. Coskun, T. S. Rosing, K. Whisnant and K. C. Gross, "Temperature-aware MPSoC scheduling for reducing hot spots and gradients," in Proc. Asia and South Pacific Design Automation Conference (ASP-DAC '08), pp. 49-54, 2008.

[12] R. Jayaseelan and T. Mitra, "Temperature aware scheduling for embedded processors," in Proc. 22nd International Conference on VLSI Design, pp. 541-546, 2009.

[13] F. Mulas, M. Pittau, M. Buttu, S. Carta, A. Acquaviva, L. Benini, D. Atienza and G. D. Michele, "Thermal balancing policy for streaming computing on multiprocessor architectures," in Proc. Design, Automation and Test in Europe Conference and Exhibition (DATE '08), pp. 734-739, 2008.

[14] A. Merkel, F. Bellosa and A. Weissel, "Event-driven thermal management in SMP systems,"in Proc. Second Workshop on Temperature–Aware Computer Systems (TACS '05) ,2005.

[15] T. Chantem, R. P. Dick and, X. S. Hu, " Temperature-aware scheduling and assignment for hard real-time applications on MPSoCs," in Proc. Design, Automation and Test in Europe Conference and Exhibition (DATE '08), pp.288-293, 2008.

[16] Dali Zhao, H. Homayoun and A. V. Veidenbaum, "Temperature aware thread migration in 3D architecture with stacked DRAM," International Symposium on Quality Electronic Design (ISQED), Santa Clara, CA, 2013, pp. 80-87.

[17] W. C. Jong Sung Lee and Kevin Skadron, "Predictive temperature-aware dvfs," IEEE Trans. Computers, pp 127-133, 2010.

[18] K. Baati and M. Auguin, "Temperature-aware DVFS-DPM for real- time applications under variable ambient temperature," Industrial Embedded Systems (SIES),8th IEEE International Symposium, vol. 13, no. 20, pp.19-21, June 2013.

[19] R. Urunuela, A. M. Deplanche and Y. Trinquet, "Storm - a simulation tool for real-time multiprocessor scheduling evaluation," in Proc. IEEE International Conference on Emerging Technology and Factory Automation, ETFA, Bilbao, pp.1-8, 2010.

[20] P. Michaud and Y. Sazeides, "ATMI: analytical model of temperature in microprocessors", pp.12-21, 2008.

[21] R. Stafford. Random vectors with fixed sum. [Online] Available http://www.mathworks.com/matlabcentral/fileexchange/9700, 2006. 98, 198.

BIOGRAPHIES OF AUTHORS

Tayyaba Bokhari received her BS degree in Computer Engineering and MS in Electrical Engineering from COMSATS Institute of Information and Technology (CIIT), Lahore, Pakistan in 2010 and 2015 respectively. She is serving University of Lahore (UOL), Lahore, as an Assistant Professor of Electrical Engineering, since March 2013. Her research interest include real-time embedded systems, context-aware computing in multicore systems, computer architecture, multi-core scheduling.

Sajjad H. Shami received his B.Sc. degree in Electrical Engineering from University of Engineering and Technology (UET), Lahore, Pakistan in 1986. He got his Ph.D. degree in Electronic Systems Engineering from University of Essex, Colchester, UK in 2000. He has nearly three decades of national and international experience in university teaching as well as academic and industrial research. He has served Motorola UK as 3G Wireless Research Engineer. He was on faculty at Northumbria University, Newcastle-upon-Tyne, UK for seven years and also at UET Lahore for eight years. Since Oct 2009, he is with University of Management and Technology (UMT), Lahore, as Professor of Electrical Engineering. His research interests include wireless networks and devices, evolutionary computing, multi-agent systems, electronic components, smart-grid and renewable energy. He has authored several research papers and also holds an international patent in 3G telecoms.

Farhan Haseeb received his BS degree in Computer Engineer from COMSATS Institute of Information Technology (CIIT), Lahore, Pakistan in 2009. He is currently serving in Exponento Pvt. Ltd., Lahore, as Staff Software Engineer. His work includes the Big Data, Databases, Cloud based system infrastructure, Application architecture, data analysis, Linux based server management and enterprise web application.