# Efficient DVFS based Energy Saving Scheme for Video Processing Applications of Mobile Systems

*[1]Yen-Lin Chen, [1]Ming-Feng Chang, [1]Wen-Yew Liang, [2]Hsin-Han Chiang, [3]Tsu-Tian Lee, [4]Meng-Tsan Li, and Chien Lin
[1]Dept. Computer Science and Information Engineering, National Taipei University of Technology Taipei, Taiwan, *ylchen@csie.ntut.edu.tw, t8599009@ntut.edu.tw, wyliang@csie.ntut.edu.tw
[2]Dept. Electrical Engineering, Fu-Jen Catholic University, New Taipei City, Taiwan, hsinhan@ee.fju.edu.tw
[3]Dept. Electrical Engineering, Tamkang University, New Taipei City, Taiwan, ttlee@ntut.edu.tw
[4]Smart Network System Institute, Institute for Information Industry, Taipei, Taiwan, holyli@iii.org.tw, stevenlin@iii.org.tw

***Abstract*—Energy efficiency, at the present time, becomes an important consideration in computer software engineering. Dynamic voltage and frequency scaling (DVFS) is an efficient approach for reducing CPU power consumption. This paper exploits DVFS technique to schedule the frame tasks to cores with higher or lower VF settings depending on a task's time-varying compute intensity. The DVFS based scheduling scheme minimizes the number of voltage and frequency transition as possible to solve the limitation of traditional DVFS for multicore systems. The proposed DVFS based scheduling scheme can improve the energy consumption by 9% to 16% with great speedup for parallel execution. This proposed scheme can effectively improve the energy saving efficiency of video processing applications for mobile computing and healthcare systems.**

***Keywords—dynamic voltage and frequency scaling (DVFS); energy saving; mobile computing; video processing; embedded systems.***

## I. INTRODUCTION

For this years, there has been a huge growing demand for high-performance computing and mobile systems with multitasking. Multicore processors are providing a solution to achieve high performance computing under power constraint. Multicore processors is a single computing component with two or more independent processing units and each core have one or more levels of a private cache. Each process core is relatively simpler and easier to design and validate than a single large SMT processor. Multiple tasks can be run separately on each core, or one task can be split into several threads and can be executed on the cores paralleling and a high throughput can be achieved without increasing the clock rate.

The use of one or more voltage regulators that deliver power to a circuit decides the behavior of the multicore DVFS technique. The DVFS technique for multicore systems can be performed at different levels of granularity including: per-chip DVFS [4], per-core DVFS [5], [6], [7], and cluster DVFS [8], [9], [10]. Per-chip DVFS (Fig. 1a) adopts the same voltage regulator to each CPU core and each CPU core applies the same voltage and frequency setting. Per-core DVFS (Fig. 1b) uses diverse voltage regulators for each core and allows the core to concurrently use different voltage and frequency settings. Cluster DVFS (Fig. 1c) takes multiple voltage regulators to drive a set of voltage/frequency islands.

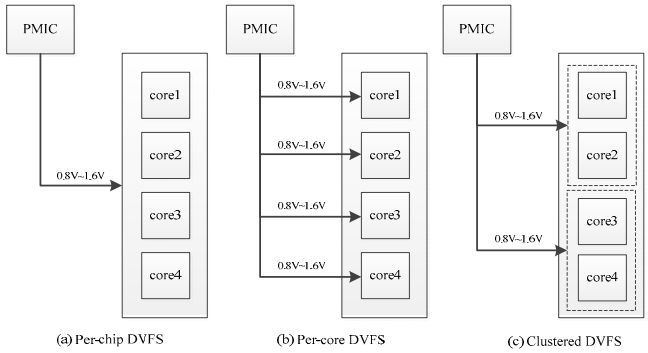


Fig. 1. Four-core MPSoC with various DVFS domains.

DVFS schemes seek to exploit runtime variability in application behavior to get the maximum energy savings with minimal performance degradation. However, conventional DVFS scaling, which is initiated by the operating system, has two crucial drawbacks [2]: First, DVFS on operating system uses the OS scheduler sampling intervals to perform computing and scaling. However, the sampling intervals for operating system are on the millisecond time scale, while computational requirements can differ on the nanosecond time scale due to events such as cache misses. Hence, OS-driven DVFS would be too slow to respond to such fine variations in program behavior. Second, multi-core systems execute multiple tasks with potentially different computational behaviors. Even though the per-core DVFS in multi-core systems, providing per-core, independent voltage control in chips with more than two cores can be expensive in reality. Moreover, when DVFS is applied across multiple cores, determining a single optimal DVFS setting that simultaneously satisfies all cores will be extremely difficult; some applications will suffer performance loss or power overheads. The

978-1-5090-1897-0/16/$31.00 ©2016 IEEE

drawback gets worsen as the number of cores and running applications increase in future systems.

Today, the growing demand of multimedia applications which are associated with higher computing complexity and, therefore, consume more power consumption, has prompted

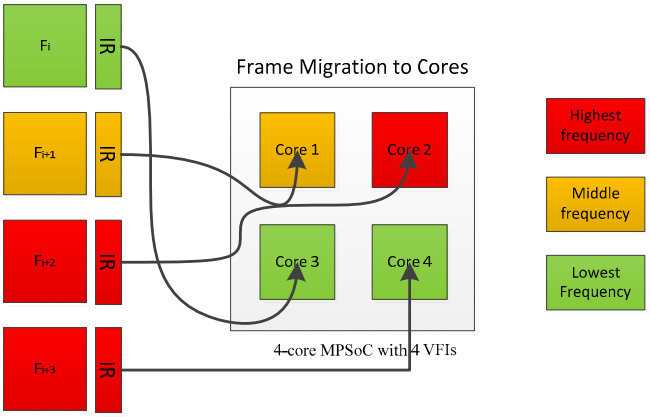


Fig. 2. DVFS based parallel scheduling.

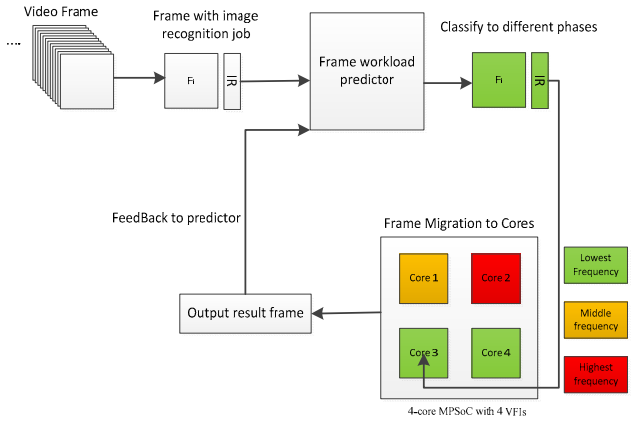


Fig. 3. Structure of the scheduling DVFS.

efforts to conserve the battery power of portable devices. The video recognition on computer system has becomes a high computing complexity and consumed huge amount of power on security system and vehicular systems. In this paper, a fast and novel DVFS power-management approach was proposed for mobile video recognition applications and healthcare systems. The proposed DVFS power-management schedules the application tasks to cores on the basis of DVFS requirement in a multi-core system with homogeneous cores.

## II. DVFS Based Parallel Scheduling Scheme

The idea of DVFS based scheduling is to schedule the frame task to an appropriate core under predetermined frequency and voltage setting. Fig. 2 presents a four core multicore system scheduling example. $F_i$ is a CPU workload low required frame task, and then we can schedule it to a lowest frequency core. Similar to $F_i$, $F_{i+1}$ and $F_{i+2}$ can schedule to core 1 and core 2 according to the frame task workload. However, $F_{i+3}$ get the same workload with $F_{i+2}$ but unfortunately left core 4 on the idle state. There are two options for scheduling frame task $F_{i+3}$. First, scheduling the task to the waiting queue on core 2. The penalty of scheduling to core 2 is load unbalance for parallel execution. Second, scheduling the task to core 4 and scaling the frequency to the highest

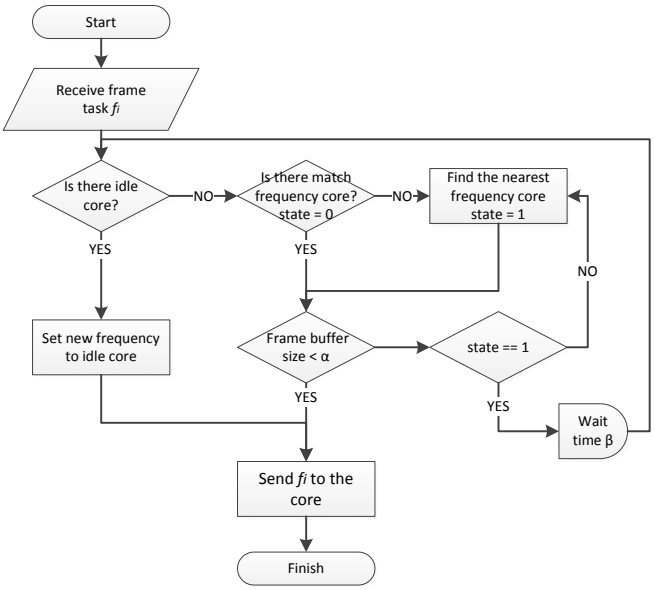


Fig. 4. The frame scheduling flow chart.

frequency. It courses some voltage and frequency transition delay on the frequency and voltage scaling. The potential benefit of scheduling task to cores is that, by doing so, DVFS no need to adjust their voltage and frequency setting every time period to save lots of transition penalty and solve the load balancing on the multicore parallel problem simultaneously.

As shown in Fig. 3, the proposed DVFS scheme divided the video recognition into one frame with image recognition task and frame workload predictor predicts the system workload of the frame task. After predicting the frame workload, the DVFS scheme classifies the frame task to different phases to supply a minimal voltage and frequency without performance degradation. Final, the DVFS scheme migrate the frame task to the core with an appropriate CPU frequency. Compared to the conventional DVFS schemes incur a large voltage and frequency transition delay, the proposed DVFS scheme reduce the number of voltage and frequency adjustments. The proposed DVFS power management with frame migration offers two important benefits: (1) it saves a large voltage and frequency transition delay to arrive at the target core with proper CPU frequency. The DVFS transition delay is due to off-chip voltage regulators that limit how quickly voltage can change, and the frequency transition delay comes from PLL relock times; (2) the DVFS scheme also provide a good quality parallel scheduling on load balancing for the video recognition.

The frame scheduling flow chart listed in Fig. 4. First, the scheduling policy checks the idle core. If there is an idle core in the system, then set the require voltage and frequency setting to the idle core and send the frame task to the core to

execution. Otherwise, the scheduling policy finds the core with matching voltage and frequency setting. If the core existed, then checking the job buffer to prevent numerous tasks schedule to one core. In the case of no matching core found, then the last choice is finding the nearest voltage and frequency setting core in all cores. If the nearest core still accumulation of a lot of tasks, then the scheduling policy stop for a time period

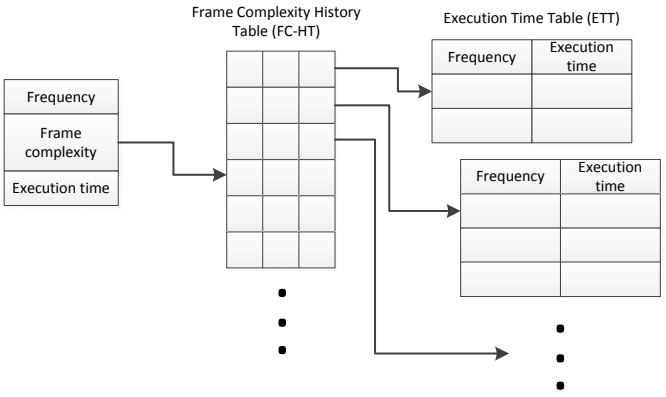


Fig. 5. Frame table predictor structure.

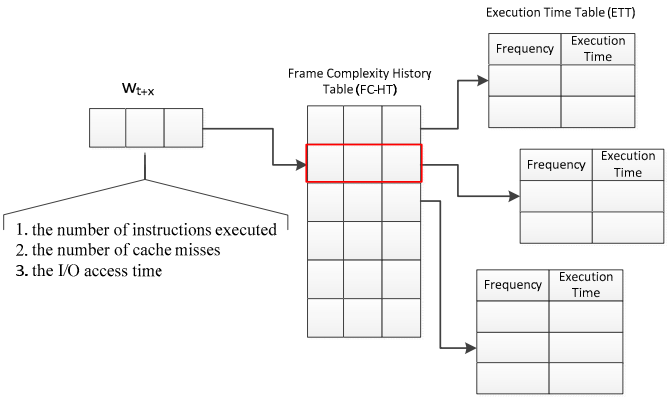


Fig. 6. Prediction of the next frequency.

for waiting other tasks to finish and restart the scheduling policy.

To predict the frame workload and corresponding frequency, the frame history table predictor in our previous work [1] is used. The table predictor retrieves the performances counters including: cache miss, instruction count, executing time and CPU frequency. The frame table predictor profiling the frame workload complexity ($W_t$), the CPU frequency ($f_t$), and the frame execution time ($E_t$) to the history table. The proposed frame predictor profiles the frame complexity-related information ($W_t$), the CPU frequency ($f_t$) and the frame execution time ($E_t$) once the recognition task finished processing a frame. The definition of frame complexity ($W$) contains three different type of system performance data: number of cache misses, the number of instructions executed and the I/O accesses. The history table for frame predictor is divided into two components, as showed in Fig. 5: First, the frame complexity history table (FC-HT) contains the executed frame performance information for the video frames. Second, the execution time table (ETT) for each of the FC-HT entry contains the applied CPU frequency and the processing time for the video frame recognition. Once a frame has been recognized, the frame predictor searches in the FC-HT table and records the frame-executing complexity information into the corresponding ETT table. If the executing complexity

TABLE I. POWER CONSUMED BY THE AMD PHENOM II IN THE IDLE STATE

| Frequency (GHz) | Idle Power (W) |
|---|---|
| 3.0 | 25.5W |
| 2.3 | 16.11W |
| 1.8 | 10.29W |
| 0.8 | 5.7W |

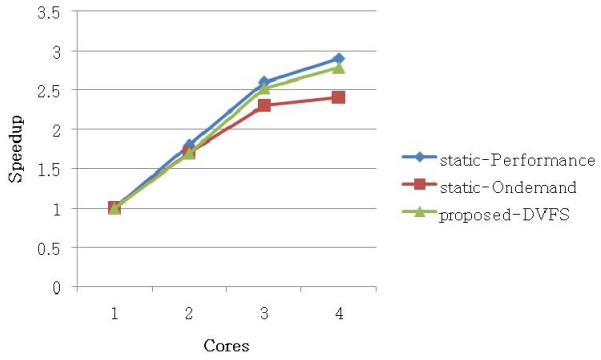


Fig. 7. Speedup results for governors.

information does not exist, then the table predictor create a new entry in FC-HT and the corresponding ETT for it.

The GPHT predictor predicts the next frame complexity $W_{t+1}$. The GPHT predictor exploits the history pattern from recent history ($W_t$ -$W_{t\text{-}depth}$) and predicts the next future frame complexity $W_{t+1}$. The commonly used statistical predictor for next pattern, the last value predictor, is assumed the last seen behavior to be the next pattern, and last value method is efficient for most of practical applications. However, the poor prediction result would be obtained for the last value predictor due to the high variability applications. By comparison on our previous study [1], the GPHT predictor can performs much more prediction accuracy than that of the last value predictor in the high motion media benchmarks. Fig. 6 shows the final step. After the frame complexity is got from hardware performance counters, the frame predictor searches the history pattern in the FC-HT and obtains the corresponding CPU frequency with the lowest execution time with lowest frequency, according to the table ETT. If the next future frame complexity $W_{t+1}$ cannot found in the FC-HT, the predictor would find the closest Euclidean distance between FC-HT and $W_{t+1}$ to select the appropriate FC-HT entry. If the required execution time in ETT are without expectations, the predictor conservatively chooses a not tested CPU frequency from largest to smallest to determine the lowest frequency that the frame can use. Lastly step, the frame table predictor uses the predicted frequency $f_{t+1}$ to the voltage regulator for DVFS adjustment.

## III. EXPERIMENTS

The experiments were performed on an AMD Phenom II X4 945 Processor on the desktop, the quad-core AMD Phenom II supports four different frequencies in Table I (0.8GHz, 1.8GHz, 2.3GHz and 3.0GHz). Each core can operate at an independent frequency. The experiments were run on an Ubuntu Linux 12.04 system with the 3.2.0-70 kernel.

| | Resolution | Type of Video | Video Length |
|---|---|---|---|
| **Inception**  | 1920*1080 | high-motion movie | 461 |
| **Shrek**  | 1920*1080 | high-motion movie | 520 |
| **Iron man**  | 1920*1080 | high-motion movie | 302 |
| **Marix**  | 1920*1080 | high-motion movie | 330 |
| **Ice Age**  | 1920*1080 | high-motion movie | 462 |
| **The Da Vinci Code**  | 1920*1080 | high-motion movie | 360 |

Fig. 8. Video benchmarks.

For comparing with the energy efficiency of parallel applications, we adopted speedup as the metric for the experiment performance. The speedup is a very appropriate metric because speedup has been the most important metric for measuring performance in parallel computing in the past decades. The speedup performance for the Inception clip is presented in Fig. 7. The static-Performance policy is using the highest CPU frequency for all cores and scheduling policy is round-robin scheduling. The static-Ondemand is using the Linux Ondemand DVFS governor with round-robin scheduling. The speedup result of the proposed DVFS is better than the static-Ondemand and closed to the static-Performance policy. The static-Ondemand used the traditional DVFS to scale the CPU speed. It scales the CPU frequency every fixed time period and cost a lot of penalty on OS scheduler and voltage transition delay.

The energy data were collected by using PowerAPI [3]. PowerAPI is a middleware toolkit for building software-defined power meters. Software-defined power meters are configurable software libraries that can estimate the power consumption of software in real-time. PowerAPI also offers an API which can be used to express request about energy spent by a process, following its hardware resource utilization (in term of CPU, memory, disk, etc.). For the video recognition, we implemented a simple motion detection technique based on image block difference and mean shift algorithm. Six high definition clips were also used as the test set for matching the user realistic usage. We used Pthread to spread our task to a specific CPU core. The pthread_setaffinity_np() function sets

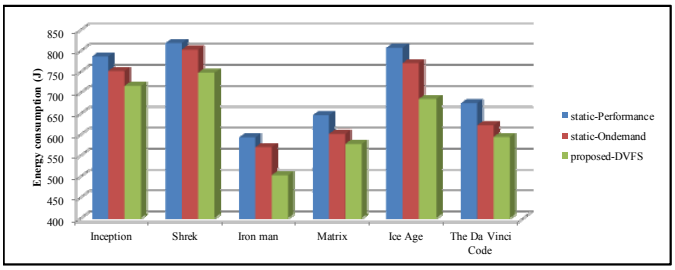


Fig. 9. Energy consumption of the clips.

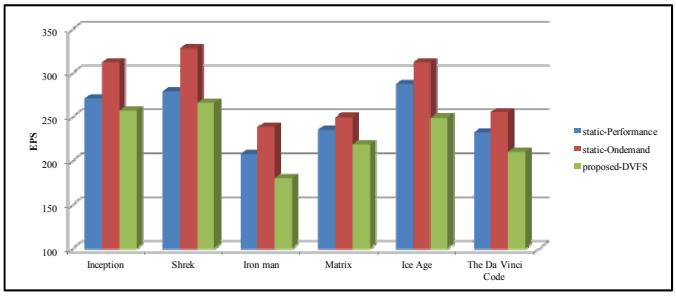


Fig. 10. Energy per speedup (EPS) results.

the CPU affinity mask of the thread to the CPU set pointed to by cpuset. Six high definition clips were used in Fig. 8.

Fig. 9 shows the energy consumption of the platform running the videos with different schemes. The static-Performance shows the worst results because it used the highest CPU frequency for all core. The static-Ondemand reduced the energy consumption by 2% to 6.9% with respect to the static-Performance scheme. By comparison, the proposed-DVFS can further improve the energy consumption by 9% to 16%. Among the experimental clips, the Iron-man clip got a great result of reducing 16% of energy consumption by the proposed-DVFS method. Other clips reduced the energy consumption by 9% to 15.2%. From these results, the proposed scheme successfully reduced the energy consumption of the videos with image processing job by as much as 16%. In other words, the proposed scheme can extend the battery usage time for 1.6 hours at most for the device that can process videos for 10 hours on multicore platform.

For comparison with the parallel and energy consumption efficiency, the paper [11] defined new metrics, Speedup per Watt (SPW), Energy per Speedup (EPS) and Energy per Target (EPT). We used the EPS to compare with other governors:

$$EPS = \frac{Power \times Time}{Speedup} \quad (1)$$

The EPS changes the focus from performance to power, and is more intuitive when energy saving is concerned, since the smallest EPS value suggests the least energy to be consumed. EPS tells us how much energy is needed for each performance unit (i.e. speedup). In the situations where performance is not a concern, we should focus on how much power is needed to achieve the targeted performance. The lower EPS represent the better result on parallel and energy efficiency. The EPS results shows in Fig. 10. The static-Ondemand gets the worst ESD results. This shows the traditional DVFS cost a lot of penalty on multicore system. The proposed-DVFS gets the best results on EPS. These results demonstrate that the proposed energy saving scheme can provide effective solution for mobile healthcare systems.

## IV. CONCLUSIONS

In this paper, a novel and original DVFS scheme for video recognition on multicore mobile computing systems using frame scheduling was proposed. The DVFS based scheduling solved the limitation and penalty of traditional DVFS for multicore systems. The proposed DVFS scheme minimizes the OS scheduling time and number of voltage transition in DVFS technique. At the same time, we parallel the frame tasks into cores based on the DVFS to achieve the best load balance on multicore system. The experimental results show that the proposed method can effectively reduce the energy consumption by 9%-16% in the clips with great parallel speedup. However, the frame dependency still needs to solve on some video recognition techniques for mobile healthcare systems. In the future work, the scheduling-based DVFS scheme for multicore system will transplant to the mobile consumer electronics devices. Simultaneously, more video recognition techniques will be tested for the DVFS scheme.

## ACKNOWLEDGMENT

This study was supported by the Ministry of Science and Technology of Taiwan with Grant. No. MOST-104-2221-E-027 -054.

## REFERENCES

[1] Yen-Lin Chen, Ming-Feng Chang and Wen-Yew Liang. "Energy-efficient video decoding schemes for embedded handheld devices," *Multimedia Tools and Applications* (2015), vol. 75, no. 6, pp. 3281-3300, 2015.

[2] K. K. Rangan, G.-Y. Wei, and D. Brooks, "Thread motion: fine-grained power management for multi-core systems," *in Proc. of the 36th annual international symposium on Computer architecture*, pp. 302-313, 2009.

[3] A. Bourdon, A. Noureddine, R. Rouvoy and L. Seinturier, "Powerapi: A software library to monitor the energy consumed at the process level," *ERCIM News*, vol. 92, 2013.

[4] S. G. Kim, H. Eom, H. Y. Yeom and S. L. Min, "Energy-centric DVFS controlling method for multi-core platforms," *Computing*, vol.96, no.12, pp. 1163-1177, 2014.

[5] Y. Rong and X. Qiang, "Learning-Based Power Management for Multicore Processors via Idle Period Manipulation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol.33, no.7, pp.1043-1055, July 2014.

[6] P. Bogdan, R. Marculescu, S. Jain, R.T. Gavila, "An Optimal Control Approach to Power Management for Multi-Voltage and Frequency Islands Multiprocessor Platforms under Highly Variable Workloads," *Proc. of Sixth IEEE/ACM International Symposium on Networks on Chip (NoCS)*, pp.35-42, 9-11 May 2012.

[7] W. Kim, M.S. Gupta, G.-Y. Wei, D. Brooks, "System level analysis of fast, per-core DVFS using on-chip switching regulators," *Proc. of IEEE 14th International Symposium on High Performance Computer Architecture (HPCA)*, pp.123-134, Feb. 2008.

[8] T. Kolpe, A. Zhai, S.S. Sapatnekar, "Enabling improved power management in multicore processors through clustered DVFS," *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp.1-6, March 2011.

[9] J. Lee, B.-G. Nam, H.-J. Yoo, "Dynamic Voltage and Frequency Scaling (DVFS) scheme for multi-domains power management," *Proc. of IEEE Asian Solid-State Circuits Conference (ASSCC)*, pp. 360-363, Nov. 2007.

[10] A. Elewi, M. Shalan, M. Awadalla, and E. M. Saad, "Energy-efficient task allocation techniques for asymmetric multiprocessor embedded systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 13, no. 2, pp. 27, Jan. 2014.

[11] J. Mair, K. Leung, Z. Huang, "Metrics and task scheduling policies for energy saving in multicore computers," *Proc. of 2010 11th IEEE/ACM International Conference on Grid Computing (GRID)*, pp.266-273, 25-28 Oct. 2010.