Enhanced PSO Approach for Real Time Systems Scheduling

Medhat Awadalla and Abdullah Elewi

***Abstract*—Systems as asymmetric multiprocessor platforms are considered power-efficient multiprocessor architectures, efficient task partitioning (assignment) and play a crucial role in achieving more energy efficiency at these multiprocessor platforms. This paper addresses the problem of energy-aware static partitioning of periodic real time tasks on heterogeneous multiprocessor platforms. A modified Particle Swarm Optimization variant based on min-min and priority assignment algorithms for task partitioning is proposed. The proposed approach aims to minimize the overall energy consumption, meanwhile avoid deadline violations. An energy-aware cost function is proposed to be considered in the proposed approach. Extensive simulated experiments and comparisons with related approaches have been conducted and the achieved results demonstrate that the proposed partitioning scheme significantly outperforms in terms of the number of executed iterations to accomplish a specific task in addition to the energy savings.**

***Index Terms*—Task partitioning, task assignment, heterogeneous multiprocessors, particle swarm optimization, min-min, priority assignment algorithm.**

I. Introduction

Nowadays, embedded systems are involved in most details of our life such as smart phones, pocket PCs, personal digital assistants (PDAs), multimedia devices, etc. As the applications on these devices are being complicated, there is a need to increase the performance while keeping the energy consumption of these devices in accepted levels especially for the portable battery-powered ones. So, minimizing energy consumption to prolong the battery life while achieving higher performance is a critical issue in the design of portable embedded systems. As the processor is one of the most important power consumers in any computing system, today's chip multiprocessor (CMP) or multiprocessor system on chip (MPSoC) platforms can deliver a higher performance at the cost of lower power consumption than uniprocessor systems. Embedded systems today are often implemented upon platforms comprised of different kinds of processing units, such as CPU's, DSP chips, graphics co-processors, math co-processors, etc., with each kind of processing unit specialized to perform a different function most efficiently. Such platforms are commonly referred to as heterogeneous platforms [1]-[3]. TI's OMAP™ [4] mobile processors are good example of these heterogeneous platforms. The multiprocessor scheduling of recurrent real-time tasks can be generally carried out under the partitioned scheme or under the global scheme. In the partitioned scheme, the tasks are statically partitioned among the processors and all instances (jobs) of a task are executed on the same processor and no job is permitted to migrate among processors. In the global scheme, a task can migrate from one processor to another during the execution of different jobs. Furthermore, an individual job of a task that is preempted from some processor, may resume execution in a different processor. Nevertheless, in both schemes, parallelism is prohibited, i.e., no job of any task can be executed at the same time on more than one processor. This paper considers the partitioned scheduling scheme. The problem of partitioning tasks among processors, sometimes, [1], [5], referred to as Task Assignment Problem (TAP), is an intractable NP-Hard problem even if the processors are homogeneous, [6]. So, approximation algorithms and heuristic techniques are used to solve this problem. This paper proposes a modified Particle Swarm Optimization (PSO) variant based on Min-min technique and priority assignment algorithm for energy-aware task partitioning on heterogeneous multiprocessor platforms. The rest of this paper is organized as follows: Section II reviews existing research on task partitioning upon heterogeneous platforms and related areas. Section III defines the problem and describes task, processor, and power models used in this paper. Section IV presents PSO, Min-min and priority assignment techniques for task partitioning and introduces our proposed approach. Section V presents simulation results for the proposed algorithm and discusses these results. Section VI summarizes our conclusions.

Manuscript received October 26, 2014; revised May 9, 2015.

Medhat Awadalla is with the Electrical and Computer Engineering Department, SQU, Oman (e-mail: medhatha@squ.edu.om).

Abdullah Elewi is with Communications, Electronics and Computers Department, Helwan University, Egypt.

II. Review Stage

The author in [7] proved that task partitioning among heterogeneous multiprocessors is intractable (strongly NP hard), represented the problem as an equivalent Integer Linear Programming (ILP) problem, and designed a 2-step approximation algorithm for solving this problem. The idea of LP relaxations to ILP problems is used in the first step to map most tasks, while in the second step the algorithm maps the remaining tasks using exhaustive enumeration. This two-step algorithm takes time polynomial in the number of tasks, and exponential in the number of processors. Braun *et. al.* [8] used tree partitioning in the second step instead of exhaustive enumeration to make the algorithm takes time polynomial in the number of tasks, and polynomial in the number of processors. They compared 11 heuristics for mapping a set of independent tasks onto heterogeneous distributed computing systems. The best one that has minimum makespan, that is defined as the maximum

DOI: 10.7763/IJCTE.2016.V8.1059

completion time for the whole processors, was the Genetic Algorithm (GA) followed by Min-min algorithm. Chen and Cheng [9] applied the Ant Colony Optimization (ACO) algorithm. They proved that ACO outperforms both GA and LP-based approaches in terms of obtaining feasible solutions as well as processing time. The authors in [5] presented a modified algorithm based on the Particle Swarm Optimization (PSO) for solving this problem and showed that his approach outperforms the major existing methods such as GA and ACO methods. Then, his PSO approach is developed to can further optimize the solution to reduce the energy consumption by minimizing average utilization of processors (without using any energy or power model). Finally, a tradeoff between minimizing the design makespan as well as energy consumption is obtained. The work in [10] presented a hybrid PSO method for solving the task assignment problem. Their algorithm has been developed to dynamically schedule heterogeneous tasks onto heterogeneous processors in a distributed setup. It considers load balancing and handles independent non-preemptive tasks. The hybrid PSO yields a better result than the normal PSO when applied to the task assignment problem. The results are also compared with GA. The results infer that the PSO performs better than the GA. Omidi and Rahmani [11] used PSO for task scheduling in multiprocessor systems as an important step for efficient utilization of resources. They considered independent tasks on homogeneous multiprocessor systems. Apart from all these efforts, this paper integrates the PSO approach with a polynomial-time partitioning techniques; Min-min and priority assignment. The proposed approach takes into account energy efficiency during task partitioning among heterogeneous cores in MPSoCs.

III. System Model

This paper considers the problem of power-aware task partitioning on heterogeneous multiprocessor platforms. So, models of task, processor, and power are presented.

A periodic real-time task τ_i generates an infinite sequence of task instances (jobs). Each job executes for C time units at most, be generated every T time units, and has a relative deadline D time units after its arrival.

This paper considers a periodic task set $\{\tau_1, \tau_2,\ldots,\tau_n\}$ of n independent real-time tasks. A task is τ_i represented as 3-tuple (C_{ij}, D_i, T_i) where C_{ij} is the Worst-Case Execution Time (WCET) of task τ_i on processor j, D is the relative deadline, and T is the period. Implicit deadlines are considered in this paper, i.e., the relative deadline is assumed to be the same as the period. Each task τ_i has a utilization $U_{ij}=C_{ij}/T_{ij}$ on processor j.

A heterogeneous multiprocessor platform with m preemptive processors based on CMOS technology is defined as $\{P_1, P_2,\ldots,P_m\}$. Also this paper considers Dynamic Voltage/Frequency Scaling (DVFS) processors that supports variable frequency (speed) and voltage levels continuously, i.e., DVFS processors can operate at any speed/voltage in its range (ideal). Of course, practical DVFS processors supports discrete speed/voltage levels (non-ideal). So, the desired speed/voltage of the ideal DVFS processor is rounded to the nearest higher speed/voltage level of the practical DVFS processor supports. The time (energy) required to change the processor speed is very small compared to that required to complete a task. It is assumed that the speed/voltage change overhead, similar to the context switch overhead, is incorporated in the task execution time. In this work, it is assumed that the processor's maximum speed (frequency) is 1 and all other speeds are normalized with respect to the maximum speed. When MPSoCs platforms are considered, there are per-core and full-chip DVFS techniques [12], [13]. In the per-core DVFS, each core operates at individual frequency/voltage, and has no operating frequency constraint. On the other hand, the practical full-chip DVFS designs restrict that all the cores in one chip operate at the same clock frequency/voltage. For each processor, the tasks are scheduled according to Earliest Deadline First (EDF) scheduling algorithm. So, a processor utilization U_j which is the sum of the utilizations of tasks assigned to this processor cannot exceed 1, i.e., $U_j=\sum u_{ij}\leq 1$.

The power consumption in CMOS circuits has two main components: dynamic and static power. The dynamic power consumption which arises due to switching activity can be represented as in [8]:

$$P_{\text{dynamic}}=C_{\text{eff}}.V_{\text{dd}}\text{^}2.f \quad (1)$$

where C_{eff} is the effective switching capacitance, V_{dd} is the supply voltage, and f is the processor clock frequency (speed) which can be expressed in terms of a constant k, supply voltage V_{dd} and threshold voltage V_{th} as follows:

$$f=k.(V_{\text{dd}}-V_{\text{th}})\text{^}2/V_{\text{dd}} \quad (2)$$

The static power consumption is primarily occurred due to leakage currents (I_{leak}) (Kong *et al.*, (2012)), and the static (leakage) power (P_{leak}) can be expressed as:

$$P_{\text{leak}}=I_{\text{leak}}.V_{\text{dd}} \quad (3)$$

When the processor is idle, a major portion of the power consumption comes from the leakage. Currently, leakage power is rapidly becoming the dominant source of power consumption in circuits and persists whether a computer is active or idle [2]. So, lowering supply voltage is one of the most effective ways to reduce both dynamic and leakage power consumption. As a result, it reduces energy consumption where the energy consumption is the power dissipated over time. For simplicity reasons, Eq. (1) is reduced to a simplified power model $P=f\text{^}3$ using normalized values where f is the processor speed (frequency). Then, a simplified energy model $E=f\text{^}2$ (using normalized values) can be used.

IV. The Proposed Approach

Before introducing our proposed approach in this paper, a background on PSO and priority assignment and min-min techniques will be presented.

A. PSO

Authors in [14] developed the PSO algorithm simulating the behavior of swarms in the nature, such as birds, fish, etc. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum

particles. PSO has been successfully applied in many scientific areas and there are many variants of the algorithm. At the beginning, a set (swarm) of random solutions (particles) is used to initialize the PSO algorithm that starts iterations looking for optimal solution. During every iteration, each particle is updated by two best values. The first one is the personal best pbest that the particle has achieved so far. The second is the global best gbest obtained by any particle in the swarm. After finding the two best values, the particle updates its velocity and position according to equations (4) and (5) respectively. The following is the typical procedure of PSO:

Initialize the Population Randomly.
DO{
For each particle.{
Calculate fitness value
If the fitness value is better than the best fitness value (pbest) then set current value as the new pbest.}
Choose the particle with the best fitness value of all particles as the gbest.
For each particle.{
Calculate new velocity:

$$V_{new}= W.V_{o1d}+C1.\ R1.\ (\text{pbest-}X)+C2.\ R2.(\text{gbest-}X) \quad (4)$$

Update particle position:

$$X_{new}=X_{old}+V_{new} \quad \}\} \quad (5)$$

Until termination criterion is met.

The random numbers $R1$ and $R2$ are generated uniformly between 0 and 1 and the constants $C1$ (self-knowledge factor) and $C2$ (social-knowledge factor) are usually in the range from 1.5 to 2.5. Finally, the inertia factor W can be fixed or varied with a decreasing value as the algorithm proceeds [11] or it may be restarted as in [5]. PSO has been applied to solve the problem of task partitioning for homogeneous and heterogeneous multiprocessors [5], [10], [11]. Considering a system consisting of m processors and n tasks. A possible solution (particle) is a vector of n elements, each element is associated to a given task. Each element takes an integer value i where $1 \le i \le m$ and represents the processor that the task is assigned to. Thus, the search space size is m^n. There are k particles in the swarm that form swarm (population) size; these particles are initialized randomly.

B. Proposed Priority Assignment Algorithm

To optimize Min-min algorithm [8], Priorities have been determined from Directed Acyclic Graph, DAG, and then assigned to the tasks in such way that the important task will be assigned to the processor that eventually leads to a better scheduling. Priority assignment algorithm flowchart is illustrated in Fig. 1. In this paper, real-time tasks are considered.

C. The Proposed Modified PSO Approach

The modified PSO approach, proposed in this paper, simply modifies the initialization step in the PSO procedure by assigning priorities for each task and then incorporating a Min-min solution (particle) in the randomly generated population. This approach gives the PSO algorithm a push to start from a good solution and then the PSO goes on trying to optimize the solution resulting in the Min-min solution in the worst case. Firstly, A cost function favoring makespan (maximum processor accumulative utilization) minimization is proposed. Then, a penalty is added to the infeasible solutions that exceed the processing capacity of any processor. In other words, the cost is represented as follows [3]:

$$\text{Cost}=\text{Max}(Uj)+\text{Penalty} \quad \text{for } j=1, 2,\ldots, m \quad (6)$$

$$\text{Penalty}=\text{Sum}(Uj>1) \quad \text{for } j=1, 2,..., m \quad (7)$$

Next, the cost function is developed to incorporate energy where the proposed PSO approach tries to find energy efficient solutions. This paper introduces an energy-aware cost function considering simplified energy model as follows:

$$\text{Cost}=\text{Sum}(Uj\text{^}2)/m+\text{Penalty} \quad \text{for } j=1, 2, \ldots, m \quad (8)$$

When applying PSO, the parameters used are the swarm size k=100, No. of iterations=100, $C1$=$C2$=2 [5], and the inertia W=1 that, according to the PSO variant used, may be fixed or may decrease linearly until reaching 0 or it may be then restarted (re-excited) to 1 to decrease linearly again. When applying PSO, the parameters used are the swarm size k=100, No. of iterations=100, $C1$=$C2$=2 [5], and the inertia W=1 that, according to the PSO variant used, may be fixed or may decrease linearly until reaching 0 or it may be then restarted (re-excited) to 1 to decrease linearly again.

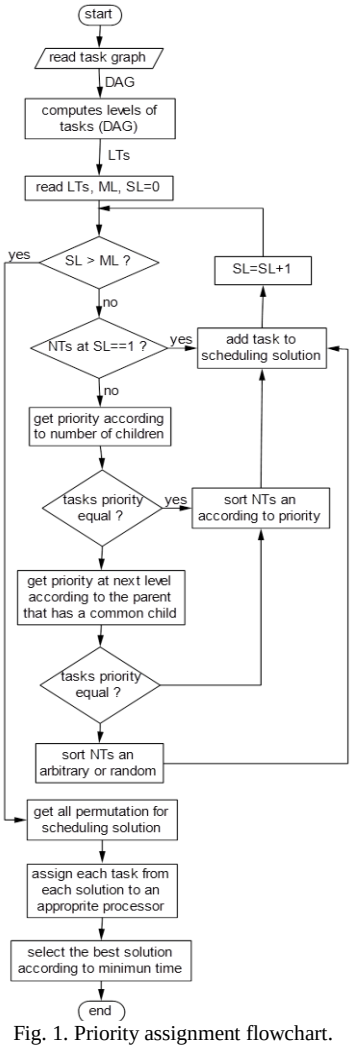

Fig. 1. Priority assignment flowchart.

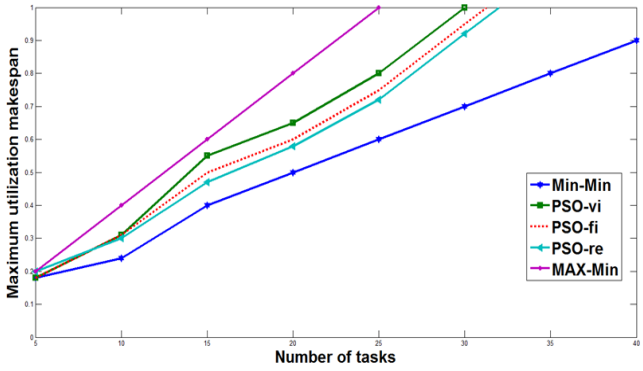

Fig. 2. A comparison of partitioning methods with light tasks partitioned upon 4 processors.

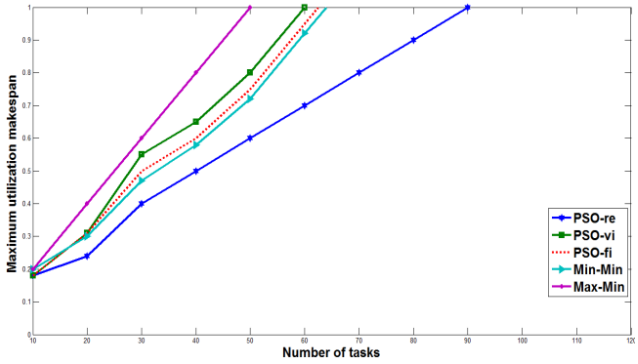

Fig. 3. A comparison of partitioning methods with light tasks partitioned upon 10 processors.

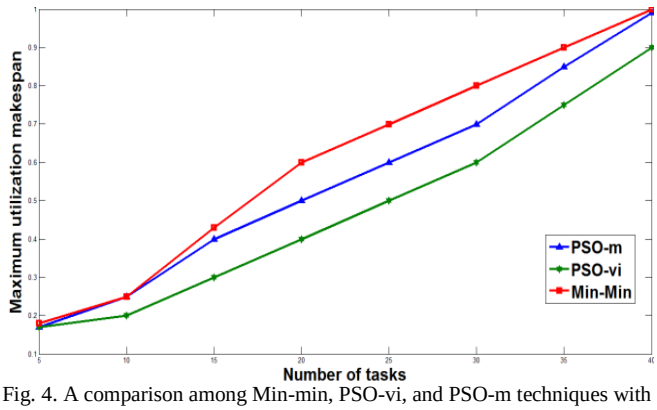

Fig. 4. A comparison among Min-min, PSO-vi, and PSO-m techniques with light tasks partitioned upon 4 processors.

The approaches have been implemented using MATLAB™. Utilization matrices have been uniformly generated of light tasks with utilization ranges from 0.05 to 0.25 and medium tasks with utilization ranges from 0.25 to 0.5. The implemented approaches are Min-min, Max-min, PSO with fixed inertia (PSO-fi), PSO with varied inertia (PSO-vi), PSO with re-excited inertia (PSOre), and our proposed Min-min based PSO approach (PSOm). With relatively small search spaces, all PSO variants show good results with reasonable number of iterations. But, when search spaces grow, so much iterations are needed to get good results using PSO approaches. PSO variants using variable inertia such as PSO-vi and PSO-re show better performance than PSO with fixed inertia (PSO-fi) with the same number of iterations and the same problem instances. Fig. 2 and Fig. 3 show comparisons among Min-min, Max-min, and PSO variants with 200 iterations for light tasks scheduled on 4 and 10 cores respectively. Our enhanced PSO gives better results with reasonable number of iterations. In the worst case, it is very close to Min-min performance if it could not optimize the solution.

Fig. 4 and Fig. 5 show the performance of our proposed approach with 100 iterations and light tasks assigned to 4 and 10 cores respectively. It is obvious that our proposed approach behaves so better when the search space grows. When medium tasks are used, the proposed approach behaves in the same way and shows better performance especially with large search spaces. Fig. 6 and Fig. 7 show the case when medium tasks are partitioned on 8 and 16 processors respectively. If per-core DVFS is considered, the introduced energy-aware cost function, Eq. (8), is taken into account. Fig. 8 and Fig. 9 show the case of partitioning light tasks on per-core DVFS platforms of 4 cores. It is clear that using makespan cost function, Eq. (6), increases the feasibility (schedulability) of the task set more than using Eq. (8) as a cost function which is more energy efficient.

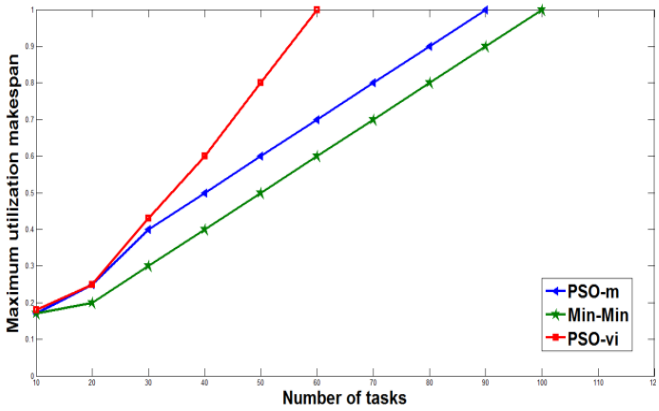

Fig. 5. A comparison among Min-min, PSO-vi, and PSO-m techniques with light tasks partitioned upon 10 processors.

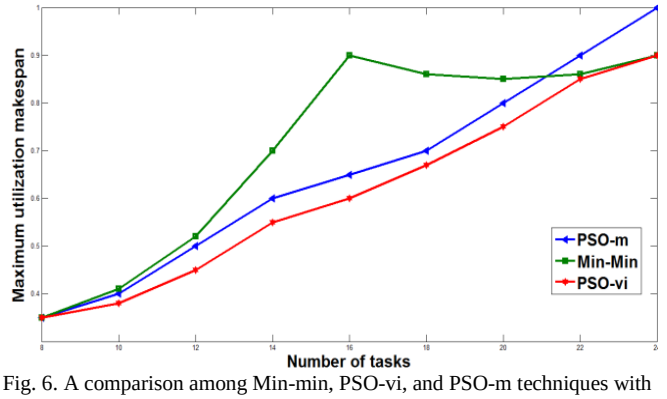

Fig. 6. A comparison among Min-min, PSO-vi, and PSO-m techniques with medium tasks partitioned upon 8 processors.

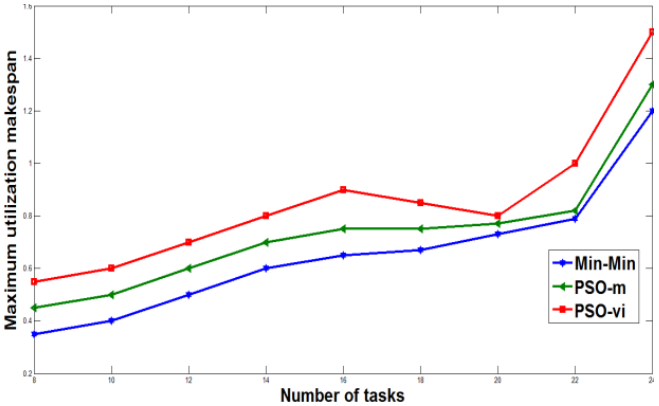

Fig. 7. A comparison among Min-min, PSO-vi, and PSO-m techniques with medium tasks partitioned upon 16 processors.

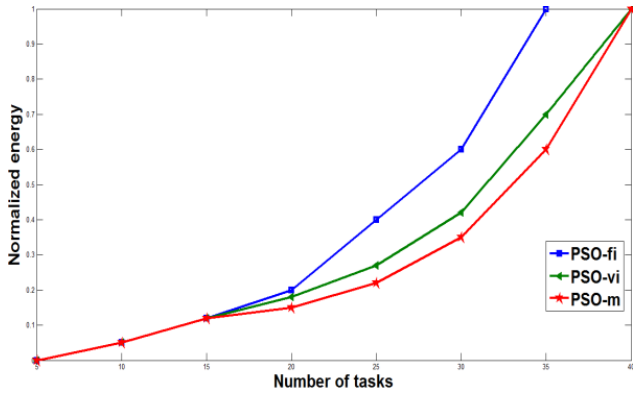

Fig. 8. A comparison among PSO-fi, PSO-vi, and PSO-m techniques with light tasks partitioned upon 4 processors.

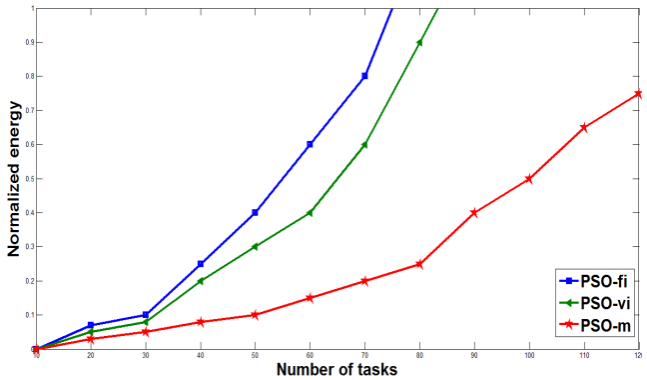

Fig. 9. A comparison among PSO-fi, PSO-vi, and PSO-m techniques with light tasks partitioned upon 10 processors.

V. Conclusions

This paper considered the problem of power-aware task partitioning on heterogeneous multiprocessor platforms. The paper proposed a modified PSO variant based on Min-min and priority assignment algorithms that outperformed its counterparts in less number of iterations for the same problem instance. Also, the energy-aware cost function is addressed in this paper and it differentiated between the full-chip and per-core DVFS processors. As a future work, any verified polynomial-time partitioning technique can be added as a particle to the population in the initialization step to give the PSO algorithm a forward push to get better solutions.

References

[1] L. Dawei and J. Wu, "Task partitioning upon energy-aware scheduling for frame-based tasks on heterogeneous multiprocessor platforms," in *Proc. 41st Int. Conf. on Parallel Processing*, 2012, pp. 430-439.

[2] F. X. Kong, W. Yi, and Q. X. Deng, "Energy-efficient scheduling of real-time tasks on cluster-based multicores," in *Proc. Design, Automation & Test in Europe Conference & Exhibition*, 2011, pp. 1-6.

[3] J.-J. Chen and L. Thiele, "Task partitioning and platform synthesis for energy efficiency," in *Proc. the 15th IEEE Int. Conf. on Embedded and Real-Time Computing Systems and Applications*, 2009, pp. 393-402.

[4] Texas Instruments (TI). OMAP™ Mobile Processors. [Online]. Available: http://www.ti.com/general/docs/gencontent.tsp?contentId=46946

[5] M. B. Abdelhalim, "Task assignment for heterogeneous multiprocessors using re-excited particle swarm optimization," in *Proc. 2008 Int. Conf. on Computer and Electrical Engineering*, 2008, pp. 23-27.

[6] H. Aydin and Q. Yang, "Energy-aware partitioning for multiprocessor real-time systems," in *Proc. the International Parallel and Distributed Processing Symposium (IPDPS)*, 2003, pp. 1-9.

[7] S. Baruah, "Partitioning real-time tasks among heterogeneous multiprocessors," in *Proc. International Conference on Parallel Processing*, Montreal, Quebec, Canada, 2004, pp. 467-474.

[8] T. D. Braun, H. J. Siegel, N. Beck *et al.*, "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," *Journal of Parallel and Distributed Computing*, vol. 61, pp. 810-837, 2001.

[9] H. Chen and A. M. K. Cheng, "Applying ant colony optimization to the partitioned scheduling problem for heterogeneous multiprocessors," *ACM SIGBED Review*, vol. 2, no. 2, 2005, pp. 11-14.

[10] P. Visalakshi and S. N. Sivanandam, "Dynamic task scheduling with load balancing using hybrid particle swarm optimization," *Int. J. Open Problems Compt. Math*, vol. 2, no. 3, pp. 475-488, 2009.

[11] A. Omidi and A. M. Rahmani, "Multiprocessor independent tasks scheduling using a novel heuristic PSO algorithm," in *Proc. the 2nd IEEE International Conference on Computer Science and Information Technology*, 2009, pp. 369-373.

[12] J. Chen and C. Kuo, "Energy-efficient scheduling for real time systems on dynamic voltage scaling (DVS) platforms," in *Proc. 13th IEEE Int. Conf.*, RTCSA, 2007, pp. 28-38.

[13] D. Koufaty, D. Reddy, and S. Hahn, "Bias scheduling in heterogeneous multicore architectures," in *Proc. the 5th ACM European Conference on Computer Systems (EuroSys)*, 2010, pp. 125-138.

[14] W. Higashino, M. A. M. Capretz, and M. B. F. Toledo, "Evaluation of particle swarm optimization applied to grid scheduling," in *Proc. 23rd IEEE WETICE Conf.*, Parma, Italy, 2014, pp. 1-6.

Medhat Awadalla was born in Cairo, Egypt. He has got his B.Sc. and M.Sc. degrees from Helwan University in 1991 and 1996 respectively. He has got his PhD degree from Cardiff University, UK, 2005 in the field of mobile robots. His research interests include multiprocessor scheduling, sensor networks and real time system.

Abdullah Elewi obtained his BSc and high studies diploma in computer engineering from Aleppo University, Syria in 2005 and 2006 respectively. He obtained his M.Sc. and PhD degrees in computer engineering from Helwan University, Egypt in 2009 and 2013. His research interests include energy-aware real-time scheduling techniques, embedded systems and real-time operating systems.