I J C T A, 8(2), 2015, pp. 733-748

Survey: Real Time Scheduling Techniques for Energy Efficient Multiprocessor

Medhat Awadalla[1,2], Dawood Al-Abri[1] and Samir Al-Busaidi[1]

Abstract: This paper surveys real-time scheduling algorithms for multiprocessor systems. The survey outlines fundamental results about multiprocessor real-time scheduling that hold independent of the scheduling algorithms employed. It provides a taxonomy of the different scheduling methods, and considers the various performance metrics that can be used for comparison purposes. A detailed review is provided covering partitioned, global, hybrid scheduling algorithms, and heuristic approaches. The survey addresses the Dynamic Voltage and Frequency Scaling (DVFS) technique that is acommonly-used for power-management wherethe clock frequency of a processor is decreased to allowa corresponding reduction in the supply voltage. This reducespower consumption, which can lead to significantreduction in the energy required for a computation, particularlyfor memory-bound workloads. It found that while DVFS is effective on the older platforms,it actually increases energy usage on the mostrecent platform, even for highly memory-bound workloads.

1. INTRODUCTION

This paper considers low-power design of real time operating system techniques for task scheduling upon multicore embedded systems. During the last two decades, much work has been done and many researchers have worked in the field ofenergy-efficient scheduling. First, the paper explores the literature briefly for energy-aware real time scheduling of independent tasks and dependent tasks sharing resources. Second, multiprocessor techniques and algorithms which are main focus of this paper are reviewed for symmetric (homogeneous), heterogeneous and asymmetric multiprocessor platforms.

2. REAL-TIME SCHEDULING OVERVIEW

The real-time tasks need an algorithm that controls the execution of these tasks on the processor according to some policy. Thus, the scheduling algorithm assigns tasks to the processor and provides an ordered list of tasks, called the planning sequence or the schedule. Scheduling algorithms and policies can be classified according to many criteria into [1-4]: Off-line or on-line scheduling according to the time of making scheduling decisions.Preemptive or non-preemptive scheduling according to the preemption capability of tasks. Best-effort or timing-fault intolerant scheduling according to the nature of time constraints.Centralized or distributed scheduling according to the nature of the scheduling platform.Off-line scheduling builds a complete planning sequence with all task set parameters. The schedule is known before task execution and can be implemented efficiently. However, this static approach is very rigid; it assumes that all parameters, including release times, are fixed and it cannot adapt to environmental changes. On-line scheduling allows choosing at any time the next task to be elected and it has knowledge of the parameters of the currently triggered tasks. When a new event occurs, the elected task may be changed without necessarily knowing in advance the time of this event occurrence. This dynamic approach provides less precise statements than the static one since it uses less information, and it has higher implementation overhead. However, it manages the unpredictable arrival of tasks and allows progressive creation of the planning sequence. Thus, on-line

[1] Electrical and computer Engineering Department, SQU, Oman

[2] Communications, Electronics and Computers Department, Helwan University, Egypt

scheduling is used to cope with aperiodic tasks and abnormal overloading.In preemptive scheduling, an elected task may be preempted and the processor allocated to a more urgent task or one with higher priority; the preempted task is moved to the ready state, awaiting later election on some processor. Preemptive scheduling is usable only with preemptive tasks. Non-preemptive scheduling does not stop task execution. One of the drawbacks of non-preemptive scheduling is that it may result in timing faults that a preemptive algorithm can easily avoid. In uniprocessor architecture, critical resource sharing is easier with non-preemptive scheduling since it does not require any concurrent access mechanism for mutual exclusion and task queuing. However, this simplification is not valid in multiprocessor architecture.

Best-effort scheduling, with soft timing constraints, uses a best effort strategy and tries to do its best with the available processors. The application may tolerate timing faults.

In timing-fault intolerant scheduling, with hard time constraints, the deadlines must be guaranteed and timing faults are not tolerated.

Centralized scheduling is implemented on a centralized architecture or on a privileged site that records the parameters of all the tasks of a distributed architecture. Scheduling is distributed when each site defines a local scheduling after possibly some cooperation between sites leading to a global scheduling strategy. In this context some tasks may be assigned to a site and migrate later.So, if periodic (static), hard, and preemptive real-time tasks are considered to be scheduled on a uniprocessor platform, the scheduling type is off-line, timing-fault intolerant, preemptive, and centralized scheduling according to the criteria mentioned earlier.

Real-time scheduling theory has traditionally focused upon the development of scheduling algorithms for feasibility analysis and run-time scheduling of a real-time system [5].

The feasibility (sometimes referred as schedulability [4, 6]) analysis determines whether all jobs can complete execution by their deadlines or not. The run-time scheduling generates schedules at run-time for systems that are deemed to be feasible (schedulable).

In other words, a real-time scheduling problem is a feasibility analysis problem (determining whether the given tasks are schedulable or not) or a run-time scheduling problem (determining the suitable scheduling algorithm and the resulting schedule).

Basic scheduling algorithms are classified according to the way of assigning priorities to tasks. If the priority is assigned to a task according to a fixed parameter, like period relative deadline, or the computation time, the algorithm is static because the priority is fixed. The priorities are assigned to tasks before execution and do not change over time. The basic algorithms with fixed-priority assignment are rate monotonic (RM) [7] and inverse deadline or deadline monotonic (DM) [8]. On the other hand, if the priority assigned to the scheduling algorithm is based on variable parameters, like absolute deadlines, it is said to be dynamic because the priority is variable. The most important algorithms in this category are earliest deadline first (EDF) [7] and least laxity first (LLF) [4].

Figure 1 shows the two categories of basic scheduling algorithms and the most important algorithms in each category.

3. POWER CONSUMPTION

The power/energy consumption is a critical issue in the design of modern microprocessors, in particular for battery-powered embedded systems where prolonging the battery life is an extremely desired goal.The power consumption in CMOS circuits can be broadly classified into two main components [9,10],dynamic power consumption which arises due to switching activity in a circuit and s*tatic* power consumption which is present due to leakage currents even when no logic operations are performed.

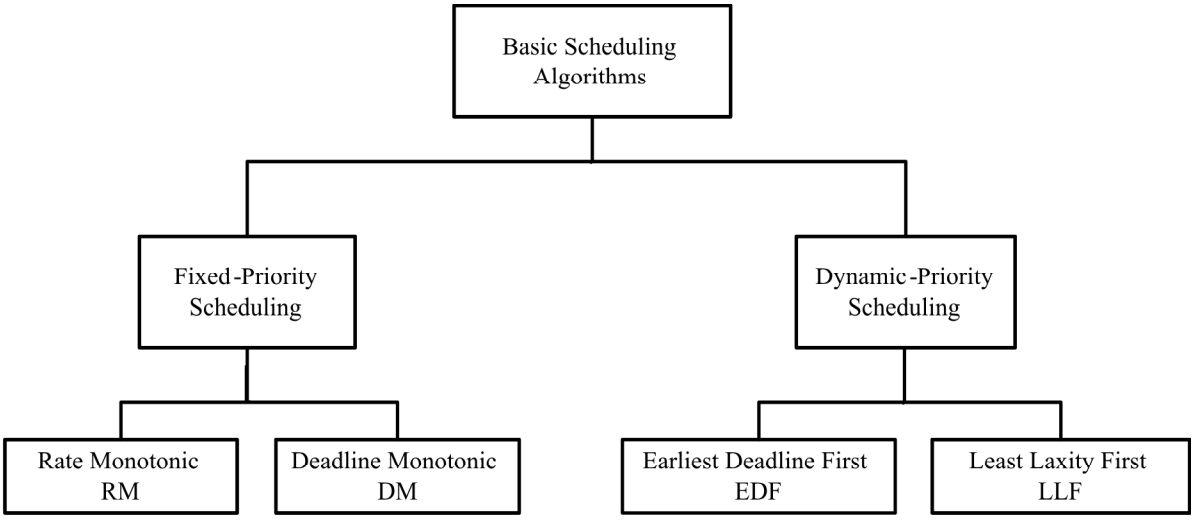

Figure 1: Basic scheduling algorithms

The power consumption has two essential components: dynamic and static power. The dynamic power consumption, which is the main component, has a quadratic dependency on supply voltage [3] and can be represented as:

$$P_{dynamic} = C_{ef} * V_{dd}^2 * F \quad (1)$$

Where C_{ef} is the switched capacitance, V_{dd} is the supply voltage, and F is the processor clock frequency (sometimes referred as speed S) which can be expressed in terms of supply voltage V_{dd} and threshold voltage V_t as following:

$$F = K * (V_{dd} - V_t)^2 / V_{dd} \quad (2)$$

The static power consumption is primarily occurred due to leakage current (I_{leak}) [3], and the static (leakage) power (P_{leak}) can be expressed as:

$$P_{leak} = I_{lead} * V_{dd} \quad (3)$$

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], and much work has been done to address this problem [3,4].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:

$$\text{Energy} = \int \text{Power dt} \quad (4)$$

$$P(s) = P_{leak} + P_{dynamic} \quad (5)$$

The total power function *P(s)* is convex and strictly increasing function with respect to speed [11, 12]. The energy is defined as the power dissipated during a certain interval of time, and the power is defined as the energy transferred per unit of time. This distinction between power and energy is important because techniques that reduce power do not necessarily reduce energy. Forexample, if the power consumed by a device to do some job can be reduced to the half, but the device then takes twice as long to do the same job, the total energy consumed will be similar. This means that the time parameter has to be taken into account in such calculations. Something else has to be mentioned here that although the total power consumption function *P(s)* is a convex and increasing function, the energy consumption function *P(s)/s* to execute a

cycle at frequency (speed) *s* is merely a convex function [11]. Hence, there exists a *critical speed*, inwhich executing any task at any speed less than the critical speed would consume more energy than that at the critical speed.

For micrometer chips, the dynamic power dominates the power consumption, while the leakage power becomes comparable for nanometer scales [11]. According to equation (1), there are four ways to reduce dynamic power consumption, though each has different tradeoffs and not all of them reduce the total energy consumed [9].The first way is to reduce the physical load capacitance or stored electrical charge of a circuit. The second way is to reduce the switching activity. The third way is to reduce the clock frequency and the fourth way is to reduce the supply voltage. According to leakage power reduction, there are four ways to reduce leakage power [9]. The first way is to reduce the supply voltage.The second way is to reduce the size of the circuit. The third way is to cool the circuit and the fourth way is to increase thethreshold voltage. As a conclusion, reducing the supply voltage which in turn requiresreducing the clock frequency (processor speed), results in reducing bothdynamic and static power consumption efficiently. This is the main goalbehind energy efficient real-time scheduling algorithms using dynamicvoltage/frequency scaling and dynamic power management techniques foruniprocessor and multiprocessor systems. Multiprocessor (multicore)systems can deliver higher performance at lower power consumption thanuniprocessor systems.

4. SYSTEM MODELS, TERMINOLOGY, AND NOTATION

This section provides a primer on the terminology and notation used even in both uniprocessor and multiprocessor scheduling research. It is aimed both at helping new researchers entering the field and providing a consistent nomenclature that has yet to fully emerge from the research community.

4.1. Uniprocessor Systems

Weiser *et al.* [13] are considered the pioneers in the field of energy efficient real-time scheduling, where they ignited the idea of the DVFS technique. Then, Yao *et al.* [14] have proposed an optimal static (offline) scheduling algorithm by considering a set of aperiodic jobs on an ideal DVFS processor. A good survey has been introduced by Chen and Kuo [11] about types and trends of energy efficient real-time scheduling techniques on dynamic voltage scaling (DVS) platforms including uniprocessor and multiprocessor systems. They categorized the related research results into eight categories in different dimensions: The first one is for aperiodic real-time jobs as in [14], while the second one is for periodic real-time tasks with dynamic priority [14] or fixed priority [15]. The third one considers energy-efficient scheduling of tasks with critical sections due to exclusive shared resources as in [16]. In [17] the optimality of EDF+SRP was shown. A dual speed energy-efficient algorithm for scheduling dependent tasks was proposed in [18]. Then extensions and developments for this algorithm were done in [19, 20, 21]. The fourth one explores systems exploiting shutdown or sleep operations when the leakage power consumption is non-negligible [22]. The fifth one reduces energy consumption by exploiting and reclaiming the slacks resulting from the early completion of jobs/tasks in an on-line fashion as in [14]. The sixth one minimizes the expected energy consumption for tasks with probability density functions of their workload information [23]. The seventh one explores the energy/power effect from I/O subsystems and other peripherals to minimize the whole system energy consumption [24]. The last one considers energy-constrained real-time scheduling which pursues performance maximization under a given energy constraint [25]. It is clearly noted that these eight categories interfere with each other and some paper can fall into many categories.

Now we will explore multiprocessor energy-efficient scheduling techniques.

4.2. Multiprocessor Systems

Classification of Multiprocessor Systems from the perspective of scheduling, multiprocessor systems can be classified into three categories. (1) Heterogeneous. The processors are different; hence the rate of execution

of a task depends on both the processor and the task. Indeed, not all tasks may be able to execute on all processors. (2) Homogeneous. The processors are identical; hence the rate of execution of all tasks is the same on all processors. (3) Uniform. The rate of execution of a task depends only on the speed of the processor. Thus a processor of speed 2 will execute all tasks at exactly twice the rate of a processor of speed 1. In this survey, we are concerned with homogeneous multiprocessor systems, comprising m processors.

4.2.1. Symmetric Multiprocessors (SMPs)

In symmetric multiprocessor (SMP) platforms, the cores (processors) arecompletely identical such as widely existed multicore processors providedby most processor vendors. The ARM MPCore™ [26] multicoreprocessor is an example of this type.A symmetric (homogeneous) multiprocessor platform with *m* preemptiveprocessors can be defined as $\Pi = \{p1, p2, \ldots, pm\}$ where *pi* represent the i^{th}processor (core).For SMP systems**,** Chen et al.[12, 27] have worked on energy efficient multiprocessor scheduling algorithms and their theoretical analysis and approximation factors. For frame-based real-time tasks with different power consumption functions, Chen and Kuo [27] proposed a 1.412-approximation algorithm. When all the cores (processors) in the system must be operated at the same speed at any time instant (full-chip DVFS), Yang, Chen, and Kuo [27] proposed a 2.371-approximation algorithm. For periodic real-time tasks, Chen *et al.* [28] proved that Algorithm Largest-Task-First (LTF) is a 1.13-approximation algorithm for periodic real-time tasks with the same power consumption characteristics. For periodic real-time tasks with different power consumption characteristics, a 1.412-approximation algorithm was proposed [29]. Chen *et al.* [28] proposed a polynomial-time 1.283-approximation algorithm for the minimization of energy consumption including leakage and the satisfaction of task timing constraints, when the overheads in turning on/off a processor are negligible, and there is no upper bound on the processor speeds. When the overheads are non-negligible, optimal solutions might require more than one processor for energy minimization, and the minimum available speed is 0, a polynomial-time 1.667- approximation algorithm is proposed to partition tasks on processors in an off-line manner and to determine the activation/dormant time in an onlinefashion. When the minimum available speed is more than 0, the proposed algorithm has a 2 approximation bound [28]. Then, by allowing tasks to run at lower speeds than the critical speed, in [12], a new approximation algorithm is developed for homogeneous multiprocessor systems with a 1.21-approximation factor, which significantly improves approximation algorithm with a 1.667- approximation factor proposed in [28]. Seo *et al*. [30] suggested dynamic repartitioning algorithm based on existing partitioning approaches for multiprocessor systems. Their algorithm dynamically balances the task loads of multiple cores to optimize power consumption during execution. They also proposed dynamic core scaling algorithm, which adjusts the number of active cores to reduce leakage power consumption under low load conditions. Wu et al. [31] proposed an energy-efficient approach to scheduling periodic real-time tasks in the multicore context. Within a voltage/frequency domain (VFD), a simple static voltage/frequency scaling schedule (Simple VS) is first introduced to select the utilization of the heaviest-loaded core as the shared operating frequency of this VFD. Next, the slack reallocation policy is proposed to further reclaim slack times while satisfying timeliness requirements. The slack reallocation strives to redistribute the slack times uniformly to the cores on the same VFD by appropriate job migrations. Lee [32] introduced two energy-saving techniques for lightly loaded multi-core processors that contain more processing cores than running tasks by exploiting overabundant cores for parallel processing of real time tasks with a lower frequency and turning off power of rarely used cores. Lee [32] also showed that the problem of minimizing total energy consumption of periodic real time tasks while meeting their deadlines is NP-hard on a lightly loaded multi-core processor. Finally, a polynomial time scheduling scheme that provides a near minimum-energy feasible schedule have been proposed. Heuristic algorithms for periodic tasks in multiprocessor environments were proposed in [33, 34] by exploiting the well-known bin-packing algorithms for task partitioning and scheduling tasks in a processor individually. Aydin and Yang [33] showed that energy aware real-time task partitioning upon identical multiprocessors problem, called *power partition*, is also an NP-hard problem that aims to partition

a list of tasks into a given number of sets that are optimally load balanced, with theguarantee that each task set can be scheduled on the system. They introduced some useful definitions and theorems. They also showed that among well-known bin-packing (BP) techniques, worst fit decreasing (WFD) generates the most balanced sets and it is the most energy efficient BP technique. Then, Alenawy and Aydin [34] addressed the energy minimization problem for fixed priority, rate monotonic (RM), scheduling and also showed its NP-hardness. In off-line settings, they showed that WFD dominates other well-known heuristics. For on-line settings, they proposed an algorithm that is based on reserving a subset of processors for light tasks to guarantee a consistent performance. Zhu *et al.* [35] proposed an algorithm in which a global queue of ready tasks is used for the selection of a candidate such that the slack time, due to the early completion of another task, is used to slow down the execution speed of the selected task. The algorithm is for a set of independent/ dependent frame-based tasks in multiprocessor environments. For dependent tasks that share resources, Lakshmanan *et al.* [36] proposed a synchronization-aware task allocation algorithm which bundles tasks that access a common shared resource and co-locate them, thereby transforming global resource sharing into local sharing and reducing the overall blocking time. Then, Nemati *et al.* [37] developed a two-round blocking-aware partitioning algorithm (BPA) which allocates tasks onto processors in a way that reduces the overall amount of blocking times of tasks. The both algorithms works under MPCP with fixed priority scheduling algorithm. They did not take energy-awareness into account and considered SMP systems. Alewi *et al.* [38] studied both static and dynamic synchronization aware energy management schemes. They devised a synchronization aware task mapping heuristic (SA-WFD) that allocates tasks accessing the same resources to the same core to effectively reduce synchronization overhead and thus improve schedulability. The SA-WFD works under an enhanced version of MSRP and partitioned EDF on SMP platforms taking energy-efficiency into account. This paper extends SA-WFD algorithm to consider AMP platforms and proposes other BP variants according to DVFS model supported by the platform.

4.2.2. Heterogeneous Multiprocessors (HMPs)

When the cores are fully different kinds of processing units such as CPUand DSP, where each kind of processing unit has a different architecture,different instruction set architecture (ISA), specialized to perform adifferent function most efficiently. Such platforms are commonly referredto as heterogeneous multiprocessor (HMP) platforms [50]. The firstgenerations of Texas Instruments (TI)'s OMAP™ [39, 40] applicationprocessors that contain DSP and ARM processors are good examples ofthis type of HMP platforms.In this paper, a heterogeneous multiprocessor platform with m preemptiveprocessors can be defined as $\Pi = \{p1, p2, \ldots, pm\}$. An independent task τ_i is represented as 3-tuple $\tau_i = (Ci,j, Di, Ti)$ where Ci,j is the worst-caseexecution time (WCET) of task τ_i on processor pj, D is the relativedeadline, and T is the period. Implicitdeadlines are considered in this paper, i.e., the relative deadline isassumed to be the same as the period. Each task τ_i has a utilization $uij = Ci,j/Ti$ on processor pj.An $n \times m$ utilization matrix as in [50] can be defined where each rowrepresents a task and each column represents a processor.

Among the work for HMP systems, Yu and Prasanna [41] considered the minimization of energy consumption for systems. The proposed algorithm is based on integer linear programming without guarantees on the schedulability of a derived solution [41]. Luo and Jha [42] proposed list scheduling- based heuristics for the scheduling of real-time tasks with precedence constraints in heterogeneous distributed systems. Genetic mapping and scheduling algorithms were developed in [43]. Hung *et al.* [44] explored energy-efficient scheduling of periodic real-time tasks in a heterogeneous system with two processing elements, in which one is a processor with the DVS capability and another is a processing element (PE) without the DVS capability. A series of approximation algorithms was proposed to minimize the energy consumption or maximize the energy saving, compared to the executions of tasks on the DVS processor. Regarding the problem of task partitioning among heterogeneous multiprocessors, Baruah [45] proved that it 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 it. 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. The same author [46] then 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. In [47], Braun *et al*. 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 completion time for the whole processors, was the Genetic Algorithm (GA) followed by Min-min algorithm. Chen and Cheng [48] 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. In [49], Abdelhalim 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. Visalakshi and Sivanandam [50] 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. In [51], Omidi and Rahmani 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. This paper considers the energy awareness of the problem of task partitioning among HMPs and proposes a PSO variant to solve it. Recently, Andersson et al.[52] considered the problem of scheduling a set of implicit deadline sporadic tasks on a heterogeneous multiprocessor platform to meet all deadlines. Tasks cannot migrate and each processor is either of type-1 or type-2 (with each task having different execution speed on each processor type). They proposed new FF variants with low time-complexity and provably good performance. Goraczko et al.[53] presented a resource model that considers the time and energy costs of run-time mode switching, which considerably improves the accuracy of existing models. Given an application, the software partitioning problem then becomes an optimization over energy cost given deadline constraints, which can be formulate as an integer linear programming (ILP) problem. They apply the resource modeling and software partitioning techniques to a multi module embedded sensing device, the Platform, and present a case study of configuring the platform for a real-time sound source localization application on a stack of MSP430 and ARM7 processor based sensing and processing boards.

4.2.3. Asymmetric Multiprocessors (AMPs)

Finally, if the cores are different in performance, power and size, but theyhave the same ISA, they are generally referred to as asymmetricmultiprocessor (AMP) platforms [39, 40]. Some authors [54],especially in the real-time community, call them uniform multiprocessors or even uniform heterogeneous multiprocessors [55]. Some authors [56]call them single-ISA heterogeneous multicore architectures while others[57] call them shared-ISA asymmetric multicore architectures but foroverlapped ISA. Furthermore, the asymmetric and heterogeneous termsmay interfere so much. In [58], heterogeneous architectures can beclassified into two types: Performance asymmetry if they have the sameISA and functional asymmetry if they have different or overlapped ISAs.Apart from all these terms and conflicts, asymmetric multiprocessors (AMP) considered in this paper have the same ISA but they are differentin performance, power and size. TI's OMAP™ application processors[39] are good example of this type. The 4th generation OMAP, OMAP4 contains dual-core ARM™ Cortex-A9 in addition to two ARM Cortex-M3s and the 5th generation OMAP,

OMAP5 SoC uses a dual-core ARM Cortex-A15 CPU with two additional ARM™ Cortex-M4 cores to offloadthe A15s in less computationally intensive tasks to increase powerefficiency.In this paper, an asymmetric multiprocessor platform with *m* preemptiveprocessors (cores) is defined as $\Pi = \{p1, p2,\ldots,pm\}$ with maximum speeds $\{S1, S2,\ldots, Sm\}$ where each processor *pj*is characterized by its relativemaximum speed *Sj*that represents the ratio between the maximum speed(frequency) of processor *pj*and the maximum speed of the slowestprocessor in the platform. The processors are ordered in non-decreasingorder of their relative maximum speeds, i.e., $1 = S_1 \leq S_2 \leq \ldots \leq S_m$. An asymmetric multiprocessor platform considered here containsprocessors that have the same ISA but they are different in performance, power and size where for some frequency $f \leq S_j$ then the processor powerfunction $P_j(f) \leq P_{j+1}(f)$ for any $1 \leq j < m$.

An independent task τ_i is represented as 3-tuple $\tau_i(C_i, D_i, T_i)$ = where C_i is the worst-case execution time of task τ_i with respect to the maximumfrequency (speed) of the slowest processor, D_i is the relative deadline, and T_i is the period. Each task τ_i has a utilization $Ui = Ci/Ti$ on the slowest processor and a utilization U_{ij} on processor *pj*, i.e., $U_{ij} = U_i/S_j$ where S_j represents the relativemaximum speed of processor P_j to the slowest processor as will be shownlater. Total utilization of all tasks (with respect to the slowest processor) is $U_{tot} = \sum_{i=1}^{i=n} u_i$.

The hyperperiod *T* of all tasks is the least common multiple of periods. The hyper period*T* of all tasks is the least common multipleof periods, i.e., $T = lcm(T_1, T_2, \ldots, T_n)$.

For AMP systems, Funk *et al.* [59] considered the on-line scheduling of hard real-time systems on uniform (asymmetric) multiprocessor machines. Resource augmentation techniques are presented here that permit online algorithms to perform better than may be expected given the inherent limitations. They explored the design of suitable on-line scheduling algorithms for use in uniform multiprocessor platforms. Then, they showed that despite its non-optimality, earliest deadline first (EDF) is an appropriate algorithm to use for on-line scheduling on uniform multiprocessors. In their work, they also presented many concepts, definitions and assumptions and developed exact and sufficient feasibility conditions for scheduling periodic real time tasks on uniform multiprocessor platforms. Baruah and Goossens [60] considered the preemptive global static-priority scheduling of systems of periodic tasks on a uniform multiprocessor platform. They obtained simple sufficient conditions for determining whether any given periodic task system will be successfully scheduled by rate monotonic (RM) upon a given uniform multiprocessor platform. Also, Funk and Baruah [55] explored partitioned scheduling on uniform multiprocessors and presented methods for finding an approximate utilization bound for partitioned scheduling on uniform multiprocessors. Then, Awadalla [61] also considered partitioned scheduling on uniform multiprocessors and proposed an algorithm that can schedule all task sets that any other possible algorithm can schedule assuming that their algorithm is given processors that are three times faster. Calandrino *et al.* [62] discussed an approach for supporting soft real time periodic tasks in Linux on performance asymmetric multicore platforms. They discussed deficiencies of Linux in supporting periodic real-time tasks, particularly when cores are asymmetric, and how such deficiencies were overcome. They also investigated how to provide good performance for non-real-time tasks in the presence of a real- time workload. For non-real-time applications, Kumar *et al.* [63] proposed and evaluated single-ISA heterogeneous (asymmetric) multi-core architectures as a mechanism to reduce processor power dissipation. Their design incorporates heterogeneous cores representing different points in the power/performance design space; during an application's execution, system software dynamically chooses the most appropriate core to meet specific performance and power requirements. Their architecture shows significant energy benefits. Li *et al.* [64] presented an asymmetric multiprocessor scheduler, AMPS, that efficiently supports both SMP and NUMA-style performance asymmetric architectures. AMPS contains three components: asymmetric aware load balancing, faster core first scheduling, and NUMA-aware migration. They have implemented AMPS in Linux kernel and used CPU clock modulation to emulate performance asymmetry on an SMP and NUMA system. For various workloads,

they showed that AMPS improves performance. Then, Li *et al.* [57] explored the design space for asymmetric multi-core architectures and presented a case study of one design, instruction-based asymmetry, in which cores have overlapping, but non-identical instruction sets. To enable transparent execution of applications, they proposed an operating system mechanism, fault-and migrate, which traps the fault when a core executes an unsupported instruction and migrates the faulting thread to one that supports the instruction. Lakshminarayana *et al.* [65] evaluated the performance of multithreaded applications in asymmetric multiprocessors. They proposed and evaluated a task size aware scheduling algorithm and a critical section length aware scheduling algorithm in asymmetric multiprocessors. The proposed scheduling algorithms simply send long task threads or long critical section threads to fast cores. They showed that when workload is asymmetric, these simple scheduling algorithms can improve performance compared to a scheduler which does not consider asymmetric characteristics. Also, Lakshminarayana and Kim [66] measured performance and power consumption in asymmetric and symmetric multiprocessors using real 8 and 16 processor systems to understand the relationships between thread interactions and performance/power behavior. They found that when the workload is asymmetric, using an asymmetric multiprocessor can save energy, but for most of the symmetric workloads, using a symmetric multiprocessor consumes less energy. Koufaty *et al.* [58] proposed bias scheduling for asymmetric multicore systems. They identified key metrics that characterize an application bias, namely the core type that best suits its resource needs. By dynamically monitoring application bias, the operating system is able to match threads to the core type that can maximize system throughput. They implemented bias scheduling over the Linux scheduler on a real system that models micro architectural differences accurately and found that it can improve system performance significantly, and in proportion to the application bias diversity present in the workload. Recently, Zhuravlev *et al.* [54] surveyed asymmetry-aware scheduling approaches in their survey of energy-cognizant scheduling techniques. They were oriented for non-real time single/multi-threaded applications. Apart from all these efforts that dealt with asymmetric (uniform) multiprocessor platforms, this paper considers energy efficiency as a main goal while partitioning real-time independent and dependent tasks upon asymmetric cores in MPSoCs taking into account the DFVS model supported by these systems.

4.3. Heuristically-based Multiprocessors scheduling

Several approaches have been adopted to solve the multiprocessor task scheduling such as heuristic approaches, evolutionary approaches and hybrid methods [70]. Authors in [71] presented a comprehensive review and classification of deterministic scheduling algorithms. Among the most common methods is a class of methods called list scheduling techniques. List scheduling techniques are widely used in task scheduling problems [72]. Insertion Scheduling Heuristic (ISH) and Duplication Scheduling Heuristic (DSH) are well-known list scheduling heuristic methods [73]. ISH is a list scheduling heuristic that was developed to optimize scheduling DAGs with communication delays. ISH extends a basic list scheduling heuristic from Hu [74] by attempting to insert ready tasks into existing communication delay slots.

The genetic-based methods have attracted a lot of researcher attention in solving the MTSP [75]. One of the main differences in these genetic approaches is their genetic operators, such as crossover and mutation. Using different crossover and mutation methods for reproducing the offspring is strongly dependent upon the chromosome representation which may lead to the production of legal or illegal solutions. Another important point in designing a GA is the simplicity of the algorithm and complexity of the evolutionary optimization process. Authors in [76] reported that the results of GA were within 10% of the optimal schedules. Their results are based on task graphs with dependencies but without communication delays. Even though the method proposed in [77] was very efficient, it does not search the entire solution space. Due to the strict ordering that only the highest priority ready task can be selected for scheduling, there can be many valid schedules omitted from the search. In [78], it is proposed some modifications to the approach in [78] to broaden the search space to include all the valid solutions. This modified approach was tested on

task graphs that represent well-known parallel programs. Authors in [79] proposed a novel GA which allows both valid and invalid individuals in the population. This GA uses an incremental fitness function and gradually increases the difficulty of fitness values until a satisfactory solution is found. This approach is not scalable to large problems since much time is spent evaluating invalid individuals that may never become valid ones. In [80], the author applies parallel GA to the scheduling problem and compares its accuracy with mathematically predicted expected value. More GA approaches are found in [81]. Another new genetic-based multiprocessor scheduling method has been presented [82]. In that paper the author claimed that the task duplication is a useful technique for shortening the length of schedules. In addition, they added new genetic operators to the GA to control the degree of replication of tasks. Some works have been performed to change the conventional approach of GA. They combined other problem solving techniques, such as divide and conquer mechanism with GA. A modified genetic approach called partitioned genetic algorithm (PGA) was proposed in [83]. In PGA, the input DAG is divided into partial graphs using a b-level partitioning algorithm and each of these separate parts is solved individually using GA. After that, a conquer algorithm cascades the subgroups and forms the final solution. In [84], a new GA called task execution order list (TEOL) was presented to solve the scheduling problem in parallel multiprocessor systems. The TEOL guarantees that all feasible search space is reachable with the same probability. Some researchers proposed a combination of GAs and list heuristics [85]. Also it proposed a modified GA by using list heuristics in the crossover and mutation in a pure genetic algorithm. This method is said to dramatically improve the quality of the solutions that can be obtained with both a pure genetic algorithm and list approach. Unfortunately, the disadvantage is that the running time is larger than running the pure genetic algorithm. Therefore the aim of this paper is to reduce that time however the modification of pure GA is done by the chromosomes' representations. Also, we grasped many ideas for designing algorithms which reduces run time and memory requirement for their implementation [86].

Baruah [87] 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. Baruah[87] 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. Braun *et al.* [88] 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 completion time for the whole processors, was the Genetic Algorithm (GA) followed by Min-min algorithm. Chen and Cheng [89] applied the Ant Colony Optimization (ACO) algorithm. They proved that ACO out performs both GA and LP-based approaches in terms of obtaining feasible solutions as well as processing time. Awadalla [90] 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. Visalakshi and Sivanandam[91] 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[92] 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.

Many Researchers have tried to implement fuzzy logic to schedule the processes. There are four main approaches reported in the literature for the fuzzy scheduling problems; fuzzifying directly the classical dispatching rules, fuzzy ranking, fuzzy dominance relation methods, and solving mathematical models to determine the optimal schedules by heuristic approximation methods [94]. Round robin scheduling using neuro fuzzy approach and Soft real-time fuzzy task scheduling for multiprocessor systems have been developed[95]. Fuzzy Better Job First (FBJF) scheduling algorithm logically integrates parameters and uses fuzzy ranking approach to determine the next most worthy job to be executed has been proposed [96]. A fuzzy scheduling approach to arrange real-time periodic and non-periodic tasks with reference to optimal utilization of distributed processors has been proposed [97]. In their paper, an attempt is made to apply fuzzy logic in the design and implementation of a modified scheduling algorithm to overcome the shortcoming of well-known scheduling algorithms. Furthermore, many dynamic and static scheduling algorithms based on fuzzy logic approach have been proposed and applied on uniprocessor systems. Also multiprocessor and distributed systems have been considered [98, 99].

5. DYNAMIC VOLTAGE AND FREQUENCY SCALING

DVFS-enabled processors have the ability to dynamically change their supply voltage and operational frequency settings during run-time of theapplication [67]. A typical DVFSenabled processor is developed, the microprocessor core carries out the required computations. This processing unit is connected through a system bus to the static memory (cache unit) and the I/O bus interface. The heart of this system, which enables a dynamic voltage selection, consists of a DC/DC voltage converter, a specialized frequency register, and a voltage controlled oscillator (VCO). Supply voltage and operational frequency are changed by writing the desired frequency *fd* into the frequency register, i.e., these changes are carried out under software control. Upon writing the desired frequency into the register, the DC/DC converter compares this frequency with the current frequency *fc* (which clocks the microprocessor core, cache, and I/O interface) and either increases or decreases the supply voltage *Vdd*. According to the changed voltage, the VCO adapts the system clock to a higher or lower frequency *fc*. Certainly, the whole voltage scaling process requires a finite time. Thus, with dynamic voltage and frequency scaling (DVFS) processors which supports many operating voltage/speed levels, energy consumption can be reduced efficiently by making appropriate decisions on the processor speed/voltage during the scheduling of real time tasks as thereis no benefit of finishing a real time job earlier than its deadline. This highlights the main idea of energy-efficient real-time scheduling ondynamic voltage/frequency scaling (DVFS) platforms.

DVFS processors have two types [11]: *ideal* and *non-ideal*. An ideal processor can operate at any speed in the range between its minimum available speed and maximum available speed. A non-ideal processor has only discrete speeds with negligible or non-negligible speed transition over heads. Another classification defines four different types of DVFS systems: ideal, feasible, practical, and multiple [68]. In this paper, it is assumed that the speed/voltage change overhead, similar to the context switch over head, is incorporated in the taskcomputation time. Also, it is assumed that the processor's maximum speed is 1 and all other speeds are normalized with respect to themaximum speed. A well-known example of DVFS processors is Intel Strong ARM SA1100 processor which runs at speeds (frequencies) ranging from 133 MHz to200 MHz. Theses frequencies are derived from two crystal oscillators, a3.6864-MHz (or 3.5795-MHz) oscillator and a 32.768-kHz oscillator [69]. The core clock frequency (speed) is configured by software throughthe core clock configuration field (CCF<4:0>) in the power-manager phase-locked-loop configuration register (PPCR). This field should be programmed during the boot sequence for the desired full-speedoperation. Table 3.1 shows oscillator frequencies as a function of the CCFsetting [69].

There are three classes of DVFS, Per-Core DVFS, Full-Chip DVFS and Per-island DVFS. In the per-core DVFS, each core operates at individual frequency/voltagelevel independently of other cores, and has no operating frequency constraint. This type of multicore is so effective as the cores operate a synchronously (independently) but at the favor of the cost.

The practical widely adopted full-chip DVFS designs restrict that all thecores in one chip to operate at the same clock frequency/voltage level oroperating performance point (OPP)[40]. In other words, the active coreson the chip have to be totally synchronous at the same OPP. Some authors [69] refer to this type as DVS-CMP. Table 3.4 shows the predefined OPPs for the TI's OMAP35x [40] application processor that contains DSP and ARM cores. Table 3.4 Operating performance points (OPPs) for the OMAP35x [40].

Recently, voltage/frequency island (VFI) [31] technique is alsoproposed. This technique allows Globally Asynchronous, LocallySynchronous (GALS) [31] design to be investigated as an alternativesolution to the totally synchronous voltage/frequency design (full-chip DVFS). Multicore (many-core) systems are partitioned into several is lands, clusters or voltage/frequency domains (VFD) [31]. In such systems, each core belongs to a specific VFI, where the active cores within the same VFI must share the same supply voltage and operating frequency. The operating frequency/voltage of each island can be adjusted independently of other domains. In other words, all cores in one island (domain) share a common voltage/ frequency while those cores between islands may operate at different frequencies. Per-island DVFS results in per-core DVFS when each VFI contains one core,while if the whole chip is just one VFI, then the full-chip DVFS is obtained. Assuming *Ni* represents the number of islands and Ncis thenumber of cores in each island, the total number of cores (processors) is*m* = Ni * Nc.

CONCLUSIONS

Currently, progress in developing multiprocessor systems is a long way ahead of research efforts to determine the best mechanisms, policies, and analysis to use in these systems. At best, this can result in systems that are heavily over specified and expensive; at worst, it can lead to intermittent and unexpected timing faults that compromise system reliability. Functionality, unit cost, time-to-market, and a reputation based on product reliability are key factors for companies developing real-time embedded systems. All of these factors can be compromised by building systems using approaches that lack the necessary theoretical underpinnings. Ultimately, multiprocessors will be used in high-integrity real-time systems, and consequently, timing failures could affect safety. Future advances along the research directions indicated in this survey should help resolve the key open issues identified. These advances hold the promise of providing the effective and efficient mechanisms, policies, and analyses required for a sound engineering-based approach to the development of complex commercial multiprocessor real-time systems.

References

[1] A. Silberschatz, P. B. Galvin and G. Gagne, Operating Systems Concepts, ninth edition, Wiley, USA, 2013.

[2] M. Awadalla, Task Mapping and Scheduling in Wireless Sensor Networks. IAENG International Journal of Computer Science, 40(4), 2013.

[3] M. Awadalla, Power Efficient Scheduling Scheme based on PSO for Real time systems. International journal of computer Applications, Volume 111, no. 4, pp. 24-30, 2015.

[4] S. Barauh and J. Goossens. "Scheduling Real-time Tasks: Algorithms and Complexity". In Handbook of Scheduling: Algorithms, Models and Performance Evaluation, J.Y.T. Leung (Eds), Chapman & Hall/CRC, 2004.

[5] A. M. K. Cheng. Real time systems: scheduling, analysis and verification, John Wiley & Sons Inc., Hoboken, New Jersey, 2002.

[6] C. Liu and J. W. Layland , Scheduling algorithms for multiprogramming in a hard real-time environment, Journal of ACM, 20(1): 46–61, 1973.

[7] J. Leung and J.W. Whitehead. On the complexity of fixed priority scheduling of periodic real-time tasks. Performance Evaluation, 2(4), 1982.

[8] G. Buttazzo. Rate-monotonic vs. EDF: Judgment day. Real-Time Systems: The International Journal of Time-Critical Computing, 29(1):5–26, 2005.

[9] V. Venkatachalam and M. Fanz. “Power Reduction Techniques For Microprocessor Systems”. ACM Computing Surveys (CSUR), Volume 37, Issue 3, pp. 195-237, 2005.

[10] F. Gruian, Energy-Centric Scheduling for Real Time Systems, PhD thesis, Lund Institute of Technology, Lund University, 2002.

[11] J. J. Chen, and C. Kuo. “Energy-efficient scheduling for real-time systems on dynamic voltage scaling (DVS) platforms”. 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2007), pp. 28-38, 2007.

[12] J.-J. Chen, and L. Thiele. “Energy-Efficient Scheduling on Homogeneous Multiprocessor Platforms”, Technical Report: TIK- 313, pp. 1-9, 2010.

[13] M. Weiser, B. Welch, A. Demers and S. Shenker. “Scheduling for reduced cpu energy”. In Proc. of Symposium on Operating system Design and Implementation (OSDI), pp. 13–23, 1994.

[14] H. Aydin, R. Melhem, D. Moss´e, and P. Mej´ýa-Alvarez. “Dynamic and aggressive scheduling techniques for power-aware real-time systems”. In Proceedings of the 22nd IEEE Real-Time Systems Symposium, pp. 95–105, 2001.

[15] S. Saewong and R. Rajkumar. “Practical voltage-scaling for fixed priority rt-systems”. In Proceedings of the 9th IEEE Real-time and Embedded Technology and Applications Symposium (RTAS’03), pp. 106–115, 2003.

[16] R. Jejurikar and R. Gupta. “Energy aware task scheduling with task synchronization for embedded real time systems”. IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, Vol. 25, No. 6, pp. 1024 – 1037, 2006.

[17] S. K. Baruah, “Resource sharing in EDF-scheduled systems: a closer look” 27th IEEE International Real-Time Systems Symposium, (RTSS’06). pp. 379–387, 2006.

[18] F. Zhang and S. T. Chanson. “Processor voltage scheduling for real time tasks with non-preemptible sections”. In Proc. of the 23rd IEEE Real-Time Systems Symposium (RTSS’02), pp. 235–245, 2002.

[19] A. M. Elewi, M. H. A. Awadalla and M. I. Eladawy. “Energy efficient real time scheduling of dependent tasks sharing resources”. In Proc. of the 2008 High Performance Computing & Simulation Conference (HPCS 2008), Nicosia, Cyprus, pp. 107-113, 2008.

[20] J. Lee, K. Koh, and C. Lee.”Multi-speed DVS algorithms for periodic tasks with non-preemptible sections”, 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications(RTCSA 2007). pp. 459-468, 2007.

[21] A. M. Elewi, M. H. A. Awadalla and M. I. Eladawy. “Energy efficient multi-speed algorithm for scheduling dependent real-time tasks”. In Proc of the 2008 Int. Conf. on Computer Engineering & Systems (ICCES’08), Cairo, Egypt, pp. 237-242, 2008.

[22] Y. Zhu and F. Mueller. “DVSleak: Combining Leakage Reduction and Voltage Scaling in Feedback EDF Scheduling”. In LCTES’07 pp.31-40, 2007.

[23] F. Gruian. “Hard real-time scheduling for low-energy using stochastic data and DVS processors”. In ISLPED, pp. 46–51, 2001.

[24] H. Cheng and S. Goddard. “Online Energy-Aware I/O Device Scheduling for Hard Real-Time Systems”. In Design, Automation and Test in Europe (DATE) , pp. 1055–1060, 2006.

[25] T. A. Alenawy and H. Aydin. “On energy-constrained real-time scheduling”. In EuroMicro Conference on Real-Time Systems (ECRTS’04), pp. 165–174, 2004.

[26] The ARM11™ MPCore™ multicore processor, http://www.arm.com/products/processors/classic/arm11/arm11-mpcore.php [last accessed 15/5/2013].

[27] J.-J. Chen and T.-W. Kuo. Multiprocessor energy-efficient scheduling for real-time tasks. In International Conference on Parallel Processing (ICPP), pp. 13–20, 2005.

[28] J.-J. Chen, H.-R. Hsu, and T.-W. Kuo. Leakage-aware energy efficient scheduling of real-time tasks in multiprocessor systems. In IEEE Real time and Embedded Technology and Applications Symposium, pp. 408–417, 2006.

[29] J.-J. Chen and T.-W. Kuo. Energy-efficient scheduling of periodic real time tasks over homogeneous multiprocessors. In the Second International Workshop On Power-Aware Real-Time Computing, pp. 30–35, 2005.

[30] E. Seo, J. Jeong, S. Park, and J. Lee, Energy efficient Scheduling of Real-Time Tasks on Multicore Processors. IEEE Trans. on Parallel and Distributed Systems 19(11), 1540–1552, 2008.

[31] X. Wu, Y. Lin, J. J. Han, and J. L. Gaudiot, “Energy-Efficient Scheduling of Real-Time Periodic Tasks in Multicore Systems”, In Network and Parallel Computing. Springer Berlin Heidelberg, pp. 344-357, 2010.

[32] W. Y. Lee, Energy-efficient Scheduling of Periodic Real-time Tasks on Lightly Loaded Multi-core Processors, IEEE Trans. on Parallel and Distributed Systems 23(3), 530–537, 2012.

[33] H. Aydin, and Q. Yang, "Energy-Aware Partitioning for Multiprocessor Real-Time Systems", In IPDPS, 2003, pp. 1-9.

[34] T. A. Alenawy and H. Aydin. Energy-aware task allocation for rate monotonic scheduling. In Proceedings of the 11th IEEE Real-time and Embedded Technology and Applications Symposium (RTAS'05), pp. 213–223, 2005.

[35] D. Zhu, R. Melhem, and B. Childers. Scheduling with dynamic voltage/speed adjustment using slack reclamation in multi-processor real time systems. IEEE Transactions on parallel and distributed systems, vol. 14, no. 7, pp. 686–700, 2003.

[36] K. Lakshmanan, D. de Niz, and R. Rajkumar. Coordinated task scheduling, allocation and synchronization on multiprocessors. In Proceedings of 30th IEEE Real-Time Systems Symposium (RTSS'09), pp. 469–478, 2009.

[37] F. Nemati, T. Nolte, and M. Behnam, "Partitioning real-time systems on multiprocessors with shared resources," In 14th Int. Conf. On Principles Of Distributed Systems (OPODIS'10), pp. 253-269, 2010.

[38] A. Elewi, M. Shalan, M. Awadalla, E. Saad"Energy-Efficient Task Allocation Techniques for Asymmetric Multiprocessor Embedded Systems", Accepted in ACM Transactions on Embedded Computing Systems (ACM TECS), Special Issue on Design Challenges for Many core Processors, 13(2), pp. 1-27, 2014.

[39] OMAP™ Application Processors. Texas Instruments (TI), http://www.ti.com/lsds/ti/omap-applications-processors/features.page. [last accessed 15/5/2013]. Computing Systems", Journal of Parallel and Distributed Computing, vol. 61, 2001, pp. 810-837.

[40] A. Musah and A. Dykstra, "Power-Management Techniques for OMAP35x Applications Processors", white paper, Texas Instruments, September 2008.

[42] J. Luo and N. Jha. Static and dynamic variable voltage scheduling algorithms for realtime heterogeneous distributed embedded systems. In the 15th International Conference on VLSI Design (VLSID'02), pp. 719–726, 2002.

[43] A. Andrei, P. Eles, Z. Peng, M. Schmitz, and B. M. Al-Hashimi. "Voltage Selection for Time-Constrained Multiprocessor Systems". In Designing Embedded Processors: A Low Power Perspective. J. Henkel and S. Parameswaran (eds.), pp. 259–284, Springer Netherlands, 2007.

[44] C.-M. Hung, J.-J. Chen, and T.-W. Kuo. Energy-efficient real-time task scheduling for a DVS system with a non-DVS processing element. In the 27th RTSS, 2006.

[45] S. K. Baruah, "Task Partitioning Upon Heterogeneous Multiprocessor Platforms", In Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'04), 2004, pp. 536 - 543.

[46] S. Baruah, "Partitioning real-time tasks among heterogeneous multiprocessors", in ICPP, Montreal, Canada, 2004, pp. 467-474.

[47] T. D. Braun, H. J. Siegel, N. Beck, L. L. Boloni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen, and R. F. Freund, "A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed.

[48] 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.

[49] M. B. Abdelhalim, "Task assignment for Heterogeneous Multiprocessors using Re-Excited Particle Swarm Optimization", In 2008 International Conference on Computer and Electrical Engineering, 2008, pp. 23-27.

[50] P. Visalakshi, and S. N. Sivanandam, "Dynamic Task Scheduling with Load Balancing using Hybrid Particle Swarm Optimization", Int. J. Open Problems Compt. Math, 2009, Vol. 2, No. 3, pp. 475 - 488.

[51] A. Omidi, and A. M. Rahmani, "Multiprocessor Independent Tasks Scheduling Using A Novel Heuristic PSO Algorithm", in Conference Name, 2009, pp. 369 – 373.

[52] B. Andersson, G. Raravi and K. Bletsas, Assigning Real-Time Tasks on Heterogeneous Multiprocessors with Two Unrelated Types of Processors, In IEEE 31st Real-Time Systems Symposium (RTSS), pp. 239-248, 2010.

[53] M. Goraczko, J. Liu, D. Lymberopoulos, S. Matic, B. Priyantha and F. Zhao, Energy-Optimal Software Partitioning in Heterogeneous Multiprocessor Embedded Systems, In Proceedings of the 45th annual Design Automation Conference (DAC'08), pp. 191-196, 2008.

[54] S. Zhuravlev, J. C. Saez, S. Blagodurov, A. Fedorova, and M. Prieto, "Survey of Energy-Cognizant Scheduling Techniques", IEEE Transactions on Parallel and Distributed Systems, 2012.

[55] S. Funk, and S. Baruah "Task assignment on uniform heterogeneous multiprocessors", in proceedings of ECRTS, 2005, pp. 219 - 226.

[56] R. Kumar, K. I. Farkas, N. P. Jouppi, P. Ranganathan, and D. M. Tullsen. Single-ISA heterogeneous multi-core architectures: The potential for processor power reduction. In Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003, pp. 81–92.

[57] T. Li, P. Brett, B. Hohlt, R. Knauerhase, S. D. McElderry, and S. Hahn "Operating System Support for Shared-ISA Asymmetric Multicore Architectures", in WIOSCA, 2008, pp. 1 - 8.

[58] D. Koufaty, D. Reddy, and S. Hahn, "Bias Scheduling in Heterogeneous Multicore Architectures", in Proceedings of the 5th ACM European Conference on Computer Systems (EuroSys), 2010, pp. 125 - 138.

[59] S. Funk, J. Goossens, and S. Baruah, "On-line scheduling on uniform multiprocessors", in proceedings of RTSS, 2001 pp. 183–192.

[60] S. Baruah and J. Goossens. "Rate-monotonic scheduling on uniform multiprocessors". IEEE Transactions on Computers, 52(7), 2003, pp. 966–970.

[61] M. Awadalla, Power Efficient Scheduling Scheme based on PSO for Real time systems. International journal of computer Applications, Volume 111, no. 4, pp. 24-30, 2015.

[62] J. Calandrino, D. Baumberger, T. Li, S. Hahn, and J. Anderson, "Soft real-time scheduling on performance asymmetric multicore platforms", in Real Time and Embedded Technology and Applications Symposium (RTAS), 2007, pp. 101 - 112.

[63] B. Andersson and E. Tovar, "Competitive Analysis of Partitioned Scheduling on Uniform Multiprocessors", in IDPDS, 2007, pp. 1 - 8.

[64] T. Li, D. Baumberger, D. A. Koufaty and S. Hahn, "Efficient Operating System Scheduling for Performance-Asymmetric Multi- Core Architectures", in Proceedings of the IEEE/ACM conference on supercomputing (SC'07), 2007, pp. 1 - 11.

[65] N. Lakshminarayana, S. Rao and H. Kim, "Asymmetry Aware Scheduling Algorithms for Asymmetric Multiprocessors", In WIOSCA'08, 2008, pp. 1 - 7.

[66] N. Lakshminarayana and H. Kim, "Understanding Performance, Power and Energy Behavior in Asymmetric Multiprocessors", In proceedings of ICCD, 2008, pp. 471 – 477.

[67] M. Schmitz, B. Al-Hashimi and P. Eles, System-Level Design Techniques for Energy-Efficient Embedded Systems, Kluwer Academic Publishers, 2005.

[69] Intel Corporation, "Intel® StrongARM® SA- Intel Corporation, "Intel® StrongARM® SA-1100 Microprocessor: Developer's Manual ", white paper, August 1999. www.lartmaker.nl/278088.pdf. [last accessed 20/5/2013].

[70] R., RahimiAzghadi,M., Hashemi, S., EbrahimiMoghadam, M., "A hybrid multiprocessor task scheduling method based on immune genetic algorithm" Qshine 2008 Workshop on Artificial Intelligence in Grid Computing 2008.

[71] Corbalan, J., Martorell, X., Labarta, J., "Performance-driven processor allocation", IEEE Transactions on Parallel and Distributed Systems, Vol. 16, No. 7, 2005, pp. 599–611.

[72] ReaKook H., Mitsuo G. and Hiroshi K., "A Performance Evaluation of Multiprocessor Scheduling with Genetic Algorithm". ReaKook Hwang et al./Asia Pacific Management Review (2006) 11(67-72).

[73] Hwang RK, Gen M., "Multiprocessor scheduling using genetic algorithm with priority-based coding", Proceedings of IEEJ conference on electronics, information and systems, 2004

[74] Montazeri, F., Salmani-Jelodar, M., Fakhraie, S.N., Fakhraie, S.M., "Evolutionary multiprocessor task scheduling", Proceedings of the International Symposium on Parallel Computing in Electrical Engineering (PARELEC'06) 2006.

[75] KamaljitKaur, AmitChhabra and Gurvinder Singh, "Modified Genetic Algorithm for Task Scheduling in Homogeneous Parallel System Using Heuristics", International Journal of Soft Computing, Vol. 5, No. 2, 2010, pp. 42-51.

[76] Wu, A.S., Yu, H., Jin, S., Lin, K.-C., Schiavone, G., "An incremental genetic algorithm approach to multiprocessor scheduling", IEEE Transactions on Parallel and Distributed Systems, Vol. 15, No. 9, 2004, pp. 824–834.

[77] G. Padmavathi and S.R.Vijayalakshmi, "Multiprocessor scheduling for tasks with priority using GA", International Journal of Computer Science Issues, IJCSI, Vol. 7, No. 1, 2010, pp. 37-42.

[78] Moore M., "An accurate parallel genetic algorithm to schedule tasks on a cluster", Parallel and Distributed Systems, Vol. 30, No. 5-6, 2004, pp. 567–583.

[79] Yao W, You J, Li B., "Main sequences genetic scheduling for multiprocessor systems using task duplication", Microprocessors and Microsystems, Vol. 28, No 2, 2004, pp. 85–94.

[80] Ceyda O, Ercan M "A genetic algorithm for multiprocessor task scheduling. In: TENCON 2004. IEEE region 10 conference, Vol. 2, 2004, pp. 68-170.

[81] Cheng S, Huang Y "Scheduling multi-processor tasks with resource and timing constraints using genetic algorithm", IEEE international symposium on computational intelligence in robotics and automation, Vol. 2, 2003, pp 624–629.

[82] Mostafa R. Mohamed, M. AWADALLA. Hybrid Algorithm for Multiprocessor Task Scheduling. IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 3, No. 2, May 2011.

[83] KamaljitKaur, AmitChhabra, Gurvinder Singh, "Heuristics Based Genetic Algorithm for Scheduling Static Tasks in Homogeneous Parallel System", International Journal of Computer Science and Security, Vol. 4, No.2, 2010, pp. 149-264

[84] Fatma A. Omara, Mona M. Arafa, "Genetic algorithms for task scheduling problem", Journal of Parallel and Distributed Computing, Vol. 70, No.1, 2010, pp. 13–22.

[85] Probir Roy, Md. MejbahUlAlam and Nishita Das, "Heuristic based task scheduling in multiprocessor systems with genetic algorithm by choosing the eligible processor", International Journal of Distributed and Parallel Systems (IJDPS) Vol. 3, No. 4, July 2012.

[86] Amit Bansal and RavreetKaur, "Task Graph Scheduling on Multiprocessor System using Genetic Algorithm", International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181, Vol. 1 Issue 5, July – 2012.

[87] Baruah S.K., (2006): Resource sharing in EDF-scheduled systems: a closer look" 27thIEEE International Real-Time Systems Symposium, (RTSS'06). pp. 379–387.

[88] Behera H. S., Pattanayak R. and Mallick P. (2012): An Improved Fuzzy-Based CPU Scheduling (IFCS) Algorithm for Real Time Systems. International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2, Issue-1, PP. 326-331.

[89] Chen J., and Kuo C. (2007): Energy-efficient scheduling for real-time systems on dynamic voltage scaling (DVS) platforms. 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2007). pp. 28-38.

[90] M. Awadalla, Energy Efficient Real Time Scheduling of Dependent Tasks. International Journal of Computers, Systems and Signals, Vol. 10, No.2, PP. 230-238, 2009.

[91] Gulati S., Arora N. and Deep K. (2012): International Journal of Research in Engineering & Applied Sciences. Volume 2, Issue 2 (February 2012), ISSN: 2249-3905, pp. 1741-1747.

[92] Hamzeh M., Fakhraie S. M. and Lucas C. (2007): Soft Real-Time Fuzzy Task Scheduling for Multiprocessor Systems. International Journal of Intelligent Technology, Volume 2, Number 4, 2007 Issn 1305-6417, Pp. 211-216.

[93] Kadhim S.J. and Al-Aubidy K.M. (2010): Design and Evaluation of a Fuzzy-Based CPU Scheduling Algorithm. V. V Das *et al.* (Eds.): BAIP 2010, CCIS 70, pp. 45 – 52, 2010.

[94] Pandey D., and Sharma M. K. (2011): Fuzzy Better Job First Scheduling Algorithm. International Journal of Computer Applications (0975 – 8887) Volume 33– No.9, PP. 13-16.

[95] Aydin, H. and Yang, Q. (2003). Energy-Aware Partitioning for Multiprocessor Real-Time Systems. In IPDPS, pp. 1- 9.

[96] Braun, T. D., Siegel, H. J., Beck, N., Boloni, L. L. Maheswaran, M. Reuther, A. I., Robertson, J. P., Theys, M. D., Yao, B., Hensgen, D. and Freund, R. F. (2001). 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, 2001, pp. 810-837.

[97] Baruah, S. (2004). Partitioning real-time tasks among heterogeneous multiprocessors. ICPP, Montreal, Quebec, Canada, pp. 467-474.

[98] Chen, H. and Cheng, A. M. K. (2005). Applying Ant Colony Optimization to the partitioned scheduling problem for heterogeneous multiprocessors. ACM SIGBED Review, Vol. 2 , No. 2, 2005, pp. 11-14.

[99] Visalakshi, P. and Sivanandam, S. N. (2009). Dynamic Task Scheduling with Load Balancing using Hybrid Particle Swarm Optimization. Int. J. Open Problems Compt. Math, Vol. 2, No. 3, pp. 475 – 48.