Energy and Synchronization-Aware Mapping of Real-Time Tasks on Asymmetric Multicore Platforms

E. M. Saad, A. M. Elewi  
Electro., Comm. and Comp. Eng. Dept., Helwan University  
Cairo, Egypt

M. Shalan  
Computer Science and Engineering Dept., AUC  
Cairo, Egypt

M. H. Awadalla  
Electrical and Computer Engineering Dept., SQU  
Muscat, Oman

ABSTRACT
Efficient task mapping plays a crucial role in saving energy in asymmetric multiprocessor platforms. This paper considers the problem of energy-aware static mapping of periodic real-time dependent tasks sharing resources on asymmetric multi/many-core embedded systems. The paper extends an existing synchronization-aware bin-packing (BP) variant when the full-chip dynamic voltage and frequency scaling (DVFS) is supported by the asymmetric multicore platform. Then, the paper proposes another BP variant when DVFS is not supported. The simulation results showed that the proposed BP variant can reduce energy consumption significantly in the presence of shared resources.

General Terms
Operating systems: Real-time embedded systems

Keywords
task partitioning; task assignment; asymmetric multiprocessors; bin-packing; shared resources; DVFS.

1. INTRODUCTION
Embedded systems are playing important roles in our lives every day. As the applications on these devices are becoming more complex, there is a need to increase the performance while keeping the energy consumption at low levels, especially for the portable battery-powered ones. Multi/many-core embedded systems can deliver higher performance while consuming lower power compared to uniprocessor systems.

Embedded systems today are often implemented using platforms comprised of multiple processing units (cores). These cores may be identical in symmetric multiprocessor (SMP) platforms, different (unrelated) in heterogeneous MP (HMP) [1] platforms or asymmetric in performance, power and size but have the same instruction set architecture (ISA) as in asymmetric MP (AMP) [2, 3] platforms. TI’s OMAP™ [4] application processors provide good examples for these different types of platforms. This paper considers asymmetric multiprocessor platforms where the processors have the same ISA but they are different in performance, size and power.

The multiprocessor real-time scheduling can be generally done under the partitioned scheme or under the global scheme [5]. In the partitioned scheme, the tasks are statically partitioned among the cores 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 on a different processor. Nevertheless, in both schemes, no job of any task can be executed at the same time on more than one processor.

This paper considers the partitioned scheme using earliest deadline first (EDF) as a uniprocessor-scheduling algorithm. The main advantage of the partitioned scheduling is that after partitioning the tasks among processors, the multiprocessor scheduling problem is reduced to a set of traditional uniprocessor ones. As the problem of partitioning tasks among multiple processors is NP-Hard [6], approximation algorithms and heuristics are used to solve this problem. Fortunately, task partitioning problem is analogous to the famous bin-packing problem (BPP) [5]. When the processors are asymmetric, it is analogous to the variable-sized BPP (VSBPP) [7].

When exclusive-access shared resources are involved in a real-time system, resource contention problems will arise. So, resource access protocols such as stack resource policy (SRP) [8] and priority ceiling protocol (PCP) [9] are needed to manage the access to shared resources while scheduling real-time tasks. Multiprocessor PCP (MPCP) [10] and multiprocessor SRP (MSRP) [11] have been also proposed.

This paper proposes BP variants for partitioning dependent tasks on asymmetric multi/many-core platforms and compares them from the energy-awareness perspective with/out DVFS. The rest of this paper is organized as follows: Section 2 reviews the related work. Section 3 defines the system model. Section 4 shows the proposed techniques. Section 5 presents simulation results. Lastly, section 6 summarizes conclusions.

2. RELATED WORK
Task mapping (assignment) on multiprocessor (multicore) systems has been explored in the literature for symmetric, heterogeneous and asymmetric multiprocessors especially with independent tasks. When tasks are dependent due to exclusive access shared resources, few researches have addressed this case especially from energy-awareness perspective.

Aydin and Yang [6] showed that energy-aware task partitioning on SMP systems problem, called power partition, is also an NP-hard problem. They showed that among well-known bin-packing heuristics, worst-fit decreasing (WFD) is the most energy-efficient one and first-fit decreasing (FFD) is the best from schedulability perspective. They built their results on symmetric multicore systems with independent tasks.
For independent tasks on AMP systems, Funk and Baruah [12] worked on utilization bounds of FFD and AFD (any-fit decreasing) that assigns each task to any processor upon which it will fit, while FMD must assign the task to the first (fastest) processor. They did not take-energy awareness into account as their goal was toward feasibility and utilization bounds. Andersson and Tovar [13] considered partitioned scheduling on uniform (symmetric) multiprocessors and proposed a FFD variant with increasing-speed ordered processors. They showed that it has a competitive factor of three, i.e., it can schedule all task sets that any other possible algorithm can schedule assuming that the algorithm is given processors that are three times faster. They did not take energy-awareness into account.

For dependent tasks that share resources, Lakshmanan et al. [14] 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. [15] 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 they considered SMP systems. Recently, Han et al. [16] proposed a synchronization-aware task allocation (SA-WFD) algorithm that allocates tasks accessing the same resources to the same core to effectively reduce synchronization overhead. 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 another BP variant when DVFS is not supported by the platform.

3. SYSTEM MODEL

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

This paper considers a periodic task set \( \Gamma = \{ \tau_1, \tau_2, ..., \tau_n \} \) of \( n \) dependent real-time tasks and a set of \( r \) serially reusable resources \( R = \{ R_1, ..., R_r \} \). A task \( \tau_i \) is represented as \( \tau_i = (D_i, \tau_i, C_i, c_{i,j,q}) \) where \( D_i \) is the relative deadline, \( \tau_i \) is the period, \( C_i \) is the worst-case execution time (WCET) of task \( \tau_i \) with respect to (w.r.t) the maximum frequency (speed) of the slowest processor, and \( c_{i,j,q} \) represents the WCET of the \( j^{th} \) critical section of task \( \tau_i \) when locks resource \( R_j \). For a task \( \tau_i \), \( C_i \) is the sum of all critical and non-critical sections. Implicit deadlines are considered in this paper, i.e., the relative deadline is assumed to be the same as the period.

Each task \( \tau_i \) has utilization \( u_i = C_i / \tau_i \) on the slowest processor and utilization \( u_{ij} \) on processor \( p_j \), i.e., \( u_{ij} = u_i / S_j \) where \( S_j \) represents the relative maximum speed of processor \( p_j \) to the slowest processor as will be shown later. The hyperperiod \( T \) of all tasks is the least common multiple of periods, i.e., \( T = lcm(T_1, T_2, ..., T_n) \).

3.2 Power and Platform Models
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 [17]:

\[
P_{\text{dynamic}} = C_{\text{eff}} V_{dd}^2 f
\]

Where \( C_{\text{eff}} \) 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 [17]:

\[
f = k(V_{dd} - V_{th})^2 / V_{dd}
\]

The static power consumption is primarily occurred due to leakage currents \( I_{\text{leak}} \) [18], and the static (leakage) power \( P_{\text{leak}} \) can be expressed as:

\[
P_{\text{leak}} = I_{\text{leak}} V_{dd}
\]

The total power consumption is the sum of dynamic and leakage power (and other types of power like short-circuit power not mentioned here). However, the total power can be considered to be composed of two parts [17]: the speed dependent part \( P_s(f) = \sum \alpha_i f^\beta \) for any \( \alpha_i > 0 \) and \( \beta > 0 \), and the speed independent part \( P_{\text{idle}} \) that may be ignored if it is small when compared to \( P_s(f) \). However, the total power function is convex and strictly increasing function with respect to speed [6, 17].

An asymmetric multiprocessor platform with \( m \) preemptive processors (cores) based on CMOS technology is defined as \( P = \{ p_1, p_2, ..., p_m \} \) with maximum speeds \( S_1, S_2, ..., S_m \) where each processor \( p_i \) is characterized by its relative maximum speed \( S_i \) that represents the ratio between the maximum speed (frequency) of processor \( p_i \) and the maximum speed of the slowest processor in the platform. The processors are ordered in non-decreasing order of their relative maximum speeds, i.e., \( f = S_1 \leq S_2 \leq ... \leq S_m \). An asymmetric multiprocessor platform considered here contains processors that have the same ISA but they are different in performance, power and size where for some frequency \( f \leq S_i \) then \( P_i(f) \leq P_{\text{idle}}(f) \) for any \( i \leq j < m \).

The paper takes into account DVFS processors that supports multiple voltage/speed levels. An ideal DFVS processor can operate at any voltage/speed level continuously, but practical DVFS processors support multiple discrete voltage/speed levels. When multi/many-core platforms are considered, there are the full-chip, per-core and per-island DVFS techniques [19]. The practical full-chip DVFS restricts all the cores in the chip to operate at the same voltage/speed level. In the per-core DVFS, each core operates at individual voltage/frequency (speed) independently of other cores, and has no operating frequency constraint. Furthermore, voltage/frequency island (VFI) technique is also proposed to get the per-island DVFS. It supports different voltage supplies and frequencies for different clusters on a multi/many-core system, where the cores on one chip can be partitioned into clusters (islands), on each of which all cores operate at a common frequency (speed). In other words, all cores in one island share a common voltage/frequency while those cores between islands may operate at different frequencies [19].

The tasks are scheduled according to EDF for each processor (core). So, a processor utilization \( U_j \) which is the sum of the
utilizations of tasks assigned to this processor must not exceed 1 else deadlines will be missed, i.e., \( U_j = \sum_{i} u_{ij} \leq 1 \).

The energy consumption of a task \( t_i \) running on a processor \( p_j \) with maximum speed \( S_j \) in an asymmetric multiprocessor platform during the hyperperiod \( T \) when executed with constant speed \( f \) \( (f \leq S_j) \) is given by: \( P(f)/(T/T_j\cdot S_j) \) [6]. Then, the energy consumption \( E_j \) of all tasks \( t_i \) allocated to the processor \( p_j \) during the hyperperiod \( T \) can be expressed as follows:

\[
E_j = \sum_{i, p_j} P(f) \cdot \frac{T}{T_j} \cdot \frac{C_i}{f} = \sum_{i, p_j} \mu_i = T \cdot P(f) \cdot \frac{U_j \cdot S_j}{f}
\]

(4)

Where \( U_j \leq 1 \) and \( U_j \cdot S_j \leq f \leq S_j \) to assure that no deadlines will be missed. Furthermore, ignoring the speed-independent power \( P_{idle} \), the minimum (optimal) energy consumption the single processor \( p_j \) consumes is when \( f = U_j \cdot S_j \) as follows:

\[
E_j^* = T \cdot P_j(U_j \cdot S_j)
\]

(5)

4. PROPOSED MAPPING TECHNIQUES

This paper extends the SA-WFD proposed in [16] to AMP systems. Before introducing the extended version of SA-WFD for AMPs, an overview of the suspension-based MSRP version considered in [16] is presented.

4.1 Suspension-Based MSRP

In [16], Han et al. considered an enhanced suspension-based version that extends MSRP [11] and OMLP [20] with the following properties:

Property 1. for local blocking time, a task can be blocked at most once by a low priority task on the same core [11].

Property 2. The local blocking time for a task is upper-bounded by the longest duration for executing a low priority task’s critical section (including the low priority task’s global waiting time, if any) on the same core [11].

Property 3. For any core, at any given time, there exists at most one task that is either a) accessing a resource; or b) suspended and waiting for a resource (which is currently accessed by a task on another core) [20].

Some definitions and notations from [16] need to be adapted to get along with AMP platforms:

\( BW \) denotes the worst-case global waiting time that can be experienced by task \( t_i \) to access all its resources.

\[
BW_i = \sum_{x=1}^{N_i^c} BW_{i,x}
\]

(6)

Where \( N_i^c \) is the number of critical sections of \( t_i \) and \( BW_{i,x} \) indicates the maximum global waiting time for task \( t_i \) assigned to core \( p_k \) when executing its \( x^{th} \) critical section

\[
BW_{i,x} = \sum_y \max\{t_j^{max}(R_y)/S_m \ | \forall t_j \rightarrow pm \}
\]

(7)

Where the arrow here means that a task on the left assigned to the core on the right, \( S_m \) is the maximum relative speed of core \( p_m \) and \( t_j^{max}(R_y) \) is the maximum amount of time for task \( t_j \) to access resource \( R_y \) once (the longest critical section of \( t_j \) that accesses \( R_y \)).

\[
t_j^{max}(R_y) = \max\{c_{j,i,a} \ | \forall x\}
\]

(8)

From Property 1, task \( t_j \) on core \( p_k \) can only be blocked at most once by a task \( t_i \) where \( t_i \) assigned to core \( p_j \) and \( T_j > T_i \). Therefore, according to Property 2, we have the maximum local blocking time for task \( t_j \) as follows:

\[
B_j = \max\{BW_j,j | \forall f : t_j \rightarrow p_k, T_j > T_i \}
\]

(9)

To adapt with AMP systems, the schedulability condition in [16] can be rewritten as follows: For a given task-to-core mapping, the tasks are schedulable under EDF on their cores if, for every core (processor) \( p_j \) there is:

\[
\forall t_j \rightarrow p_j \quad \frac{B_j + \sum_{\forall i \rightarrow p_j, j \rightarrow i} (C_i / S_j) + BW_{i,x}}{T_i}
\]

(10)

4.2 The Proposed Synchronization-Aware Techniques

The SA-WFD [16] considers values such as pessimistic estimated utilization (peu) of a task \( t_i \) that incorporates its maximum global waiting time \( BW_{i,x} \). For AMP systems, the \( pea_{ik} \) of a task \( t_i \) on a core \( p_k \) with maximum relative speed \( S_i \) can be rewritten as follows:

\[
pea_{ik} = \frac{(C_{i} / S_i) + BW_{i,x}^{max}}{T_i}
\]

(11)

\[
BW_{i,x}^{max} = \sum_{x=1}^{N_i^c} BW_{i,x}^{max}
\]

(12)

\[
BW_{i,x}^{max} = \sum_{\forall t_j \rightarrow p_k, j \rightarrow i} \frac{t_j^{max}(R_y)}{S_i}
\]

(13)

Where \( \Theta_j \) contains up to \((m-1)\) other tasks that access the resource \( R_y \) that is accessed by \( x^{th} \) critical section of task \( t_i \) and have the longest access time. That is, whenever task \( t_j \) accesses its resource \( R_y \), it is assumed to wait for other tasks, up to \((m-1)\), on different cores \( p_l \) \( (1 \leq l \leq m \text{ and } l \neq k) \) to access \( R_y \) for the longest time. If no other task accesses \( R_y \) then \( \Theta_{ij} \) is empty and \( BW_{i,x}^{max} = 0 \). The overall estimated utilization of a core \( p_k \) is defined as:

\[
EU_{i} = \sum_{\forall r_i \rightarrow p_k, j \rightarrow i} pea_{ik}
\]

(14)

Also, the resource similarity \( w_{ij} \) between two tasks \( t_i \) and \( t_j \) is defined as the number of resources that are accessed by both tasks \( t_i \) and \( t_j \). The overall resource similarity between task \( t_i \) and core \( p_k \) is defined as follows:

\[
\Omega_k(i) = \sum_{\forall r_i \rightarrow p_k, j \rightarrow i} W_{i,j}
\]

(15)

The utilization for each core \( p_k \) is defined as follows:
15. For each task

\[ U_k = \max_{\forall \tau_i \rightarrow p_k} \left\{ \frac{B_i}{T_i} + \sum_{\forall \tau_j \rightarrow p_k \cap \tau_i} \frac{(C_j / S_j) + BW_j}{T_j} \right\} \] (16)

The SA-WFD [16] is adapted to AMP systems by computing values such as pessimistic estimated utilization (\(peu_k\)) for each task \(\tau_i\) and core \(p_k\). Then, a task is assigned to the core with maximum overall resource similarity or to the core with minimum estimated utilization behaving like WF manner. As soon as the task is assigned to some core, the estimated utilization of that core has to be updated. Then, after assigning all tasks, the actual utilization of each core is computed. Thus, an asymmetric version of SA-WFD which is still valid for SMP systems is obtained. Algorithm 1 shows this AMP version of SA-WFD.

**Algorithm 1. SA-WFD for AMPs**

**Input:** A task set \(\tau = \{\tau_1, \tau_2, ..., \tau_n\}\) and a set of processors \(\{p_1, p_2, ..., p_m\}\) with maximum speeds \(\{S_1, S_2, ..., S_m\}\) where \(S_1 \leq \ldots \leq S_m\)

**Output:** \(U_1, U_2, ..., U_m\) utilizations of processors.

1. for (each task \(\tau_i\) and core \(p_k\)) do
   2. calculate \(BW_i^{\text{max}}\) from (12);
   3. calculate \(peu_k\) from (11);
2. end for
4. Sort tasks in non-increasing order of \(peu_i\);
5. for (each task \(\tau_i\) in the above sorted order) do
6.   7. Find core \(p_k\) with the maximum \(\Omega(i)\) (if more cores have the same maximum \(\Omega(i)\), the core with the minimum \(EU_i + peu_i\) is chosen; if there is still a tie, the core with smallest index is chosen);
8.   if \((EU_i + peu_i) \leq \max\{EU_j \mid 1 \leq j \leq m\}\) then
9.     Assign task \(\tau_i\) to core \(p_k\);
10. else
11.   Find core \(p_k\) with the minimum \((EU_i + peu_i)\) (if there is a tie, the core with smallest index is chosen);
12.   Assign task \(\tau_i\) to core \(p_k\);
13. end if
end for
7. for each task \(\tau_i\) calculate \(BW_i\) and \(B_i\) from (6) and (9) respectively;
8. for each core \(p_k\) calculate \(U_k\) from (16);

The important feature of WFD is that it balances the workload among cores. So, it is the most energy-efficient bin-packing technique as shown in [6] on SMP systems. With AMP systems, this result is not always true. It is true with platforms support full-chip DVFS.

This paper considers AMP platforms that support full-chip DVFS or do not support DVFS at all assuming that unused or idle cores are shut down.

**4.2.1 Full-Chip DVFS (FC-DVFS)**

When the full-chip DVFS is supported, the energy consumed by the \(m\)-core AMP platform during the hyperperiod \(T\) can be expressed as:

\[ E_{fc-dvfs} = T \sum_{j=1}^{m} (U_j / U_{\text{max}}) P_j (S_j / U_{\text{max}}) \] (17)

Where \(U_{\text{max}}\) is the maximum processor utilization among processors assuming ideal DVFS processors.

According to (17), balancing workload among cores results in minimizing \(U_{\text{max}}\) and reducing energy consumption consequently. Thus, SA-WFD is the proposed energy-efficient technique for task mapping on AMP systems when FC-DVFS is supported.

**4.2.2 Without DVFS (NO-DVFS)**

When DVFS is not supported, the energy consumed by the \(m\)-core AMP platform during the hyperperiod \(T\) can be expressed as:

\[ E_{no-dvfs} = T \sum_{j=1}^{m} U_j P_j (S_j) \] (18)

**Algorithm 2. SA-FFD for AMPs**

**Input:** A task set \(\tau = \{\tau_1, \tau_2, ..., \tau_n\}\) and a set of processors \(\{p_1, p_2, ..., p_m\}\) with maximum speeds \(\{S_1, S_2, ..., S_m\}\) where \(S_1 \leq \ldots \leq S_m\)

**Output:** \(U_1, U_2, ..., U_m\) utilizations of processors.

1. for (each task \(\tau_i\) and core \(p_k\)) do
2.   3. calculate \(BW_i^{\text{max}}\) from (12);
3.   4. calculate \(peu_i\) from (11);
4. end for
5. Sort tasks in non-increasing order of \(peu_i\);
6. for (each task \(\tau_i\) in the above sorted order) do
7.   8. Find core \(p_k\) with the maximum \(\Omega(i)\) (if more cores have the same maximum \(\Omega(i)\), the core with the minimum \(EU_i + peu_i\) is chosen; if there is still a tie, the core with smallest index is chosen);
8.   if \((EU_i + peu_i) \leq 1\) then
9.     Assign task \(\tau_i\) to core \(p_k\);
10. else
11.   Find first (smallest index) core \(p_k\) with \((EU_i + peu_i) \leq 1\);
12.   Assign task \(\tau_i\) to core \(p_k\);
13. end if
end for
15. for each task \(\tau_i\) calculate \(BW_i\) and \(B_i\) from (6) and (9) respectively;
16. for each core \(p_k\) calculate \(U_k\) from (16);
According to (18), assigning tasks to the slowest cores and saving the fastest cores will reduce the energy consumption. This paper proposes a BP variant, called SA-FFD, that emulates the SA-WFD but behaves like FF manner by assigning tasks to the first (slowest) core that fits. The processors are indexed where the slowest processor has the lowest index and so on. Algorithm 2 shows SA-FFD for AMP systems.

5. SIMULATION RESULTS

The SA-WFD and SA-FFD techniques have been implemented using MATLAB\textsuperscript{TM}. Task utilization values have been randomly (uniformly) generated to get tasks with utilization less than 0.25 or 0.5; all utilizations are considered with respect to the slowest processor. The number of resources ranges from 2 to 10. The number of critical sections of a task is 0 or 1. The critical section ratio (CSR) ranges from 0.01 to 0.10 of the WCET of a task. The asymmetric multi/many-core platforms shown here are the 4-core platform $P_4$ with maximum relative speeds $\{1,2,3,4\}$ and the 8-core platform $P_8$ with speeds $\{1,1,1,1,2,2,2,2\}$. For simplicity reasons, power consumption model implemented here is a simplified power model $P(f) = \alpha f^3$ using normalized (relative) values where $f$ is the processor speed (frequency) and $\alpha$ is a processor-dependent constant. The faster the processor is the bigger the constant $\alpha$ is. The processor $p_j$’s constant $\alpha$ is assumed to be equal to its maximum relative speed $S_j$.

Very extensive experiments and multiple runs have been done on different platforms to verify the proposed techniques. Figures 1 and 2 show the normalized energy consumption when tasks with utilization less than 0.25 are partitioned among cores on $P_4$ platform with full-chip DVFS and without DVFS respectively.

The SA-WFD is the most energy-efficient algorithm when full-chip DVFS is supported. When DVFS is not support, it is clear that the proposed SA-FFD outperforms SA-WFD. Figures 3 and 4 show the same behavior when the comparisons are done with task utilization less than 0.5 and $P_8$ as a target platform.
Generally, when DVFS is not supported the energy consumption can be reduced by 80% using SA-FFD compared to SA-WFD with light workloads as SA-WFD starts with the fastest cores while SA-FFD starts with the slowest cores. With heavy workloads, the SA-FFD consumes more energy and it approaches to SA-WFD when the cores approximate to be fully utilized.

Platforms and processor maximum speeds have a role in energy consumption. The less speed differences are, the smoother the curves are.

6. CONCLUSION

This paper addressed the problem of energy and synchronization-aware partitioning of periodic real-time dependent tasks due to shared resources on asymmetric multi/many-core embedded systems.

The paper extended the existing SA-WFD algorithm to asymmetric multicore systems and proposed SA-FFD algorithm and showed that it is the most energy-efficient technique when DVFS is not supported on AMP platforms. The simulation results showed that when DVFS is not supported, the SA-FFD could reduce the energy by 80% compared to SA-WFD with light workloads in the presence of shared resources. As a future work, other DVFS models such as per-core and per-island models will be taken into account.

7. REFERENCES