University of Nebraska - Lincoln

DigitalCommons@University of Nebraska - Lincoln

CSE Conference and Workshop Papers

Computer Science and Engineering, Department
of

2012

Adaptive Energy-Efficient Task Partitioning for Heterogeneous Multi-Core Multiprocessor Real-Time Systems

Shivashis Saha

University of Nebraska-Lincoln, ssaha@cse.unl.edu

Jitender S. Deogun

University of Nebraska-Lincoln, jdeogun1@unl.edu

Ying Lu

University of Nebraska-Lincoln, ying@unl.edu

Follow this and additional works at: <https://digitalcommons.unl.edu/cseconfwork>

Part of the [Computer Sciences Commons](#)

Saha, Shivashis; Deogun, Jitender S.; and Lu, Ying, "Adaptive Energy-Efficient Task Partitioning for Heterogeneous Multi-Core Multiprocessor Real-Time Systems" (2012). *CSE Conference and Workshop Papers*. 198.

<https://digitalcommons.unl.edu/cseconfwork/198>

This Article is brought to you for free and open access by the Computer Science and Engineering, Department of at DigitalCommons@University of Nebraska - Lincoln. It has been accepted for inclusion in CSE Conference and Workshop Papers by an authorized administrator of DigitalCommons@University of Nebraska - Lincoln.

Adaptive Energy-Efficient Task Partitioning for Heterogeneous Multi-Core Multiprocessor Real-Time Systems

Shivashis Saha, Jitender S. Deogun, and Ying Lu

Department of Computer Science and Engineering,

University of Nebraska-Lincoln, Lincoln, NE 68588-0115, U.S.A.

Email: {ssaha,deogun,ylu}@cse.unl.edu

Abstract—The designs of heterogeneous multi-core multiprocessor real-time systems are evolving for higher energy efficiency at the cost of increased heat density. This adversely effects the reliability and performance of the real-time systems. Moreover, the partitioning of periodic real-time tasks based on their worst case execution time can lead to significant energy wastage.

In this paper, we investigate adaptive energy-efficient task partitioning for heterogeneous multi-core multiprocessor real-time systems. We use a power model which incorporates the impact of temperature and voltage of a processor on its static power consumption. Two different thermal models are used to estimate the peak temperature of a processor. We develop two feedback-based optimization and control approaches for adaptively partitioning real-time tasks according to their actual utilizations. Simulation results show that the proposed approaches are effective in minimizing the energy consumption and reducing the number of task migrations.

Keywords—Adaptive Task Partitioning; Thermal-Constrained Task Partitioning; Energy Minimization; Heterogeneous Multi-Core Multiprocessor Real-Time Systems

I. INTRODUCTION

Energy-efficient designs of computer systems have received significant interest in recent years due to an increased need for energy conservation [1]. Heterogeneous multiprocessor real-time systems are known to be energy efficient and have better performance as compared to homogeneous systems [2]. The energy efficiency of recent multiprocessor systems is achieved by increasing the power density. This in turn results in high heat density which can significantly impact the reliability and performance of heterogeneous real-time systems [3].

The power consumption of a processor is divided into static and dynamic power consumption. The power consumed by the processor to maintain its activeness is called the static power consumption [1]. Similarly, the power needed by the processor while executing a task is called the dynamic power consumption. The static power is generated by the leakage current while the dynamic power is a function of the speed of the processor [4]. This function is known to be a strictly convex and monotonically increasing function and is represented by a polynomial of at least second degree [5]. This convex relationship is exploited by the *Dynamic Voltage Scaling* (DVS) techniques for minimizing the total energy consumption of a processor [6]. There has been a significant research in energy-aware scheduling for homogeneous multiprocessor systems which have negligible static

power consumption [7]. It has been shown that static power consumption is significant and is comparable to dynamic power consumption [8]. Leakage aware scheduling strategies for heterogeneous systems have been recently investigated [1], [9]. There has also been a recent interest in temperature-aware multiprocessor scheduling strategies for minimizing the temperature of a processor and thus improving its reliability [10], [11], [12]. In an ongoing project, we have been investigating thermal-constrained energy-efficient partitioning for heterogeneous multi-core multiprocessor real-time systems. Worst case execution time (WCET) of tasks are generally known to be pessimistic estimates. Thus, WCET based task partitioning may result in significant energy wastage as the processors may be unnecessarily running at high speeds due to the pessimistic estimation. Thus, there is a need for an adaptive approach for partitioning tasks in order to minimize the energy consumption in real-time systems.

In this paper, we investigate adaptive thermal-constrained energy-efficient partitioning of periodic tasks in heterogeneous multi-core multiprocessor real-time systems. We consider a system which is heterogeneous across multiprocessors, but homogeneous within a multiprocessor. Thus, our objective is to find an optimal set of active cores and partitioning of the tasks based on their actual utilization that results in minimum energy consumption while satisfying processor constraints and meeting task deadlines. We use a power model that considers the impact of temperature [12] and voltage [11] of a processor on its leakage power consumption. Two thermal models, *heat-independent thermal (HIT) model* and *heat-dependent thermal (HDT) model*, are used for estimating the peak temperature of a processor. We consider negligible heat transfer among the cores in HIT model [11], and non-negligible heat transfer in HDT model [10], [12]. We present *Distributed Utilization Control* (DUC) and *Greedy Utilization Control* (GUC) heuristics for adaptive task partitioning which are feedback-based optimization and control approaches. Simulation results show the effectiveness of the proposed heuristics in minimizing energy consumption and reducing the number of task migrations.

The rest of the paper is organized as follows: Related work is given in Section II. The system models and the problem statement are given in Section III. The adaptive task partitioning algorithms are presented in Section IV. In Section V, we present and discuss the simulation results. Finally, conclusion and future works are given in Section VI.

II. RELATED WORK

Real-time scheduling of periodic tasks in multiprocessor environment is well investigated [13]. The existing techniques can be broadly classified into *global* and *partitioning-based* scheduling techniques. In global scheduling techniques, the global scheduler selects tasks for execution primarily from a queue of tasks [14]. However, in partitioning-based scheduling techniques the processors are independently scheduled and each task is assigned to a single processor for execution [15]. Partitioning-based scheduling techniques are more widely used than global scheduling techniques due to their simplicity in design and implementation [16]. Thus, our heuristics are partitioning-based scheduling techniques aimed to minimize energy consumption while satisfying the thermal constraints.

There has been a significant amount of research in energy-aware scheduling strategies for homogeneous multiprocessor systems [7], [14]. However, energy-aware scheduling strategies for heterogeneous multiprocessor systems is still in its early stages [1]. Most of these existing work is based on power models with negligible static power consumption [17]. However, a power model with non-negligible static power consumption was recently investigated [1]. Recent research has shown that leakage current of a processor changes super linearly with its temperature [18]. Thus, thermal aware scheduling strategies in real-time systems have been recently investigated which assume dynamic leakage current [10], [11], [12], [19]. Thermal models were also proposed to consider non-negligible heat transfer between different cores in a multi-core system [10], [12]. Moreover, it has also been shown recently that leakage current is not only impacted by the temperature of a processor but also by its supply voltage [11].

Most of the partitioning-based approaches use information of WCET of tasks which is overly pessimistic and hard to predict accurately. Thus, there is a need for adaptive partitioning of tasks based on their actual execution time. Control theory has been used for developing adaptive centralized scheduling for multiprocessor real-time systems [20]. Probabilistic distributions of execution time of tasks have also been used for developing adaptive scheduling strategies [21].

III. SYSTEM MODELS AND PROBLEM DEFINITION

In this section, we describe our models and the problem.

A. Multiprocessor Model

Let $\Omega = \{M_1, M_2, \dots, M_m\}$ denotes a set of interconnected heterogeneous multiprocessor units. Each multiprocessor unit has k identical cores (or processors), i.e. $M_i = \{M_{i,1}, M_{i,2}, \dots, M_{i,k}\}$ ($i = 1, \dots, m$). A core $M_{i,j}$ ($j = 1, \dots, k$) supports dynamic voltage scaling (DVS) and varies its voltage/speed/frequency $f_{i,j}$ to one of the discrete levels in the range $[f_i^{\min}, f_i^{\max}]$, where $f_i^{\max}$ ($f_i^{\min}$) is the maximum (minimum) operating frequency of the multiprocessor unit M_i . Thus, our system is heterogeneous across multiprocessors, but homogeneous within a multiprocessor. For simplicity, the frequency of a core is normalized with respect to (wrt) $f_i^{\max}$, i.e., we assume $f_i^{\max} = 1$. The throughput (or capacity) of a core is proportional to its operating frequency [22]. The

capacity of a core $M_{i,j}$ is denoted by $\mu_{i,j}$ and is equal to $\alpha_i f_{i,j}$, where α_i is the performance coefficient of M_i . In a heterogeneous multiprocessor system, higher values of α_i correspond to more powerful multiprocessor units.

B. Task Model

Let $\Gamma = \{\tau_1, \tau_2, \dots, \tau_n\}$ denotes a set of independent periodic real-time tasks. A periodic task τ_i ($i = 1, \dots, n$) is an infinite number of task instances (jobs) released with periodicity P_i [23]. Thus, the relative deadline of a current instance (job) of τ_i is represented by its period P_i . W_i denotes the worst-case execution time of τ_i on a core of a standard multiprocessor ϕ with performance coefficient $\alpha_\phi = 1$ and it is equal to $\frac{W_i}{f_\phi}$, if the core ϕ is running at a constant frequency f_ϕ . The worst-case utilization of τ_i under the maximum frequency of a standard core is denoted by u_i and it is equal to $\frac{W_i}{P_i}$. Thus, the total worst-case utilization of task set Γ under the maximum frequency of a standard core is $U_{tot} = \sum_{i=1}^n u_i = \sum_{i=1}^n \frac{W_i}{P_i}$. The *necessary* condition for having a feasible schedule of Γ on Ω is to have $U_{tot} \leq \sum_{j=1}^m k\alpha_j$. We make this assumption throughout the paper.

Given a partitioning of tasks into cores, the worst-case utilization of $M_{i,j}$ ($i=1, \dots, m; j=1, \dots, k$) under its maximum frequency is denoted by $U_{i,j}$. If $\Gamma_{i,j}$ denotes the set of tasks allocated to $M_{i,j}$, then $U_{i,j} = \frac{\sum_{\tau_r \in \Gamma_{i,j}} W_r / P_r}{\alpha_i} = \frac{\sum_{\tau_r \in \Gamma_{i,j}} u_r}{\alpha_i}$, where $r=1, \dots, n$. As each task is allocated to exactly one core, $U_{tot} = \sum_{r=1}^n u_r = \sum_{i=1}^m \sum_{j=1}^k \alpha_i U_{i,j}$. P is the hyper-period of task set Γ , i.e. the minimum positive number such that the jobs are released every P time units. If $P_1, \dots, P_n$ are integers, then P is the least common multiple (LCM) of all task periods.

In addition to the task period P_i and worst-case execution time W_i of a task τ_i ($i = 1, \dots, n$), let the actual execution time of the q^{th} job of τ_i be $c_{i,q}$. Thus, under a constant frequency f_ϕ , the actual execution time of q^{th} job of τ_i is $\frac{c_{i,q}}{f_\phi}$. Therefore, the actual utilization of τ_i under maximum frequency ($f^{\max}=1$) on a standard core ϕ is $\tilde{u}_i = \frac{c_{i,q}}{P_i}$. Given a partitioning of tasks into cores, the actual utilization of $M_{j,r}$ ($j=1, \dots, m; r=1, \dots, k$) under its maximum frequency is $\tilde{U}_{j,r} = \frac{\sum_{\tau_i \in \Gamma_{j,r}} \tilde{u}_i}{\alpha_j}$. Thus, we aim to minimize energy consumption in hyper-period P while satisfying all the constraints.

C. Power Model

The power consumed by a core $M_{i,j}$ is given by $\Phi_{i,j}$ ($i=1, \dots, m; j=1, \dots, k$). It is composed of two parts: $\Phi_{i,j}^s$ and $\Phi_{i,j}^d$ (Eq. 1a). $\Phi_{i,j}^s$ is the static (or leakage) power consumption of a core which is generated by the leakage current for maintaining the activeness of the core [1], [8]. $\Phi_{i,j}^d$ is the dynamic power consumption of a core required for executing a task [4]. Eq. 1b gives the static power consumption of a core which is dependent on both frequency [10], [12] (proportional to the supply voltage [24]) and temperature [11], [24] of the core. γ_i , δ_i , and χ_i are non-negative constants dependent on the architecture of M_i . The dynamic power consumption of a core is a function of its frequency. Using current DVS technologies, $g(f_{i,j})$ is assumed to be a strictly convex and monotonically increasing function which is represented by a

polynomial of at least second degree. We assume $\Phi_{i,j}^d$ is a cubic polynomial in frequency (i.e. $f_{i,j}^3$) [10], [11], [12] and is given by Eq. 1c.

$$\Phi_{i,j}(f_{i,j}) = \Phi_{i,j}^s(f_{i,j}) + \Phi_{i,j}^d(f_{i,j}) \quad (1a)$$

$$\Phi_{i,j}^s(f_{i,j}) = \gamma_i f_{i,j} + \delta_i f_{i,j} T_{i,j} \quad (1b)$$

$$\Phi_{i,j}^d(f_{i,j}) = \chi_i f_{i,j}^3 \quad (1c)$$

D. Temperature Model

In this section, we present two different thermal models, *heat-independent thermal model* [11] and *heat-dependent thermal model* [10], [12].

1) *Heat-Independent Thermal (HIT) Model*: In this model we assume there is negligible or no heat transfer among cores of a multiprocessor unit or among different multiprocessors [11], [24]. Using the RC thermal model [1], [10], [11], [12], [24], the temperature of a core $M_{i,j}$ ($i = 1, \dots, m; j = 1, \dots, k$) wrt time is denoted by $T_{i,j}(t)$ and is given by Eq. 2, where T^{amb} is the ambient temperature (in $^{\circ}C$), R_i is the thermal resistance of M_i (in $J/^{\circ}C$), C_i is the thermal capacitance of M_i (in $Watt/^{\circ}C$), $\Phi_{i,j}(t)$ is the power consumption of $M_{i,j}$ wrt time (in $Watt$), and $\frac{dT_{i,j}(t)}{dt}$ is the derivative of the temperature of $M_{i,j}$ wrt time.

$$R_i C_i \frac{dT_{i,j}(t)}{dt} + T_{i,j}(t) - R_i \Phi_{i,j}(t) = T^{amb} \quad (2)$$

If initial temperature of $M_{i,j}$ at time t_0 is $T_{i,j}^0$ and $M_{i,j}$ is running at a constant frequency $f_{i,j}$, then the final temperature of $M_{i,j}$ at time t_1 is denoted by $T_{i,j}^1$ and is given by Eq. 3.

$$\frac{dT_{i,j}}{dt} = \frac{T^{amb} + \gamma_i R_i f_{i,j} + \delta_i R_i f_{i,j} T_{i,j} + \chi_i R_i f_{i,j}^3}{R_i C_i} - \frac{T_{i,j}}{R_i C_i} \quad (3a)$$

$$\int_{T_{i,j}^0}^{T_{i,j}^1} \frac{dT_{i,j}}{\frac{T^{amb} + \gamma_i R_i f_{i,j} + \chi_i R_i f_{i,j}^3}{R_i C_i} - \left(\frac{1 - \delta_i R_i f_{i,j}}{R_i C_i}\right) T_{i,j}} = \int_{t_0}^{t_1} dt \quad (3b)$$

$$T_{i,j}^1 = \frac{T^{amb} + \gamma_i R_i f_{i,j} + \chi_i R_i f_{i,j}^3}{1 - \delta_i R_i f_{i,j}} - \left(\frac{T^{amb} + \gamma_i R_i f_{i,j} + \chi_i R_i f_{i,j}^3}{1 - \delta_i R_i f_{i,j}} - T_{i,j}^0 \right) e^{-\left(\frac{1 - \delta_i R_i f_{i,j}}{R_i C_i}\right)(t_1 - t_0)} \quad (3c)$$

If $M_{i,j}$ is running at a constant frequency, then its temperature is a non-decreasing function (Eq. 3) [10], [11], [12], [24]. The temperature of $M_{i,j}$ becomes steady when the system reaches steady state condition. The peak temperature of $M_{i,j}$ running at constant frequency $f_{i,j}$ is denoted by $T_{i,j}^*$ $\left(\frac{d^2 T_{i,j}(t)}{dt^2} \Big|_{T_{i,j}(t)=T_{i,j}^*} > 0\right)$ and is given by Eq. 4.

$$T_{i,j}^* = \frac{\frac{T^{amb} + \gamma_i R_i f_{i,j} + \chi_i R_i f_{i,j}^3}{R_i C_i}}{\frac{1 - \delta_i R_i f_{i,j}}{R_i C_i}} = \frac{T^{amb} + \gamma_i R_i f_{i,j} + \chi_i R_i f_{i,j}^3}{1 - \delta_i R_i f_{i,j}} \quad (4)$$

2) *Heat-Dependent Thermal (HDT) Model*: In this model we assume non-negligible amount of heat transfer among the cores of a multiprocessor unit and negligible or no heat transfer among different multiprocessor units [10], [12]. We also assume that there is a set of heat sinks $\Xi_i = \{\varpi_{i,1}, \varpi_{i,2}, \dots, \varpi_{i,h}\}$ for each multiprocessor unit M_i ($i=1 \dots m$). These heat sinks are only used for heat dissipation and are placed on top of the cores. These heat sinks do not

generate any power. Fourier's laws can be used to model the dynamic heat transfer between the core and heat sinks of a multiprocessor unit where each core acts as a discrete thermal element [10], [12]. Using RC thermal model [10], [12], let the thermal conductance between two cores $M_{i,j}$ and $M_{i,j'}$ is $\omega_{j,j'}^i$ ($\forall j, j' \in 1 \dots k$). $\zeta_{j,q}^i$ denotes the vertical thermal conductance between core $M_{i,j}$ and sink $\varpi_{i,q}$ ($q=1 \dots h$). The horizontal thermal conductance between the heat sinks $\varpi_{i,q}$ and $\varpi_{i,q'}$ ($\forall q, q' \in 1 \dots h$) is denoted by $\omega_{q,q'}^i$. We assume that $\omega_{j,j'}^i = \omega_{j',j}^i$, $\omega_{j,j}^i = 0$, $\omega_{q,q'}^i = \omega_{q',q}^i$, and $\omega_{q,q}^i = 0$. ω^{amb} denotes the thermal conductance between the heat sink and the environment. The thermal capacitance of M_i and Ξ_i are denoted by C_i and C_i^{sink} respectively. Eq. 5 gives the temperature of $M_{i,j}$ wrt time which is denoted by $T_{i,j}(t)$, where $\frac{dT_{i,j}(t)}{dt}$ and $\Phi_{i,j}(t)$ are respectively the derivative of temperature of $M_{i,j}$ and power consumption of $M_{i,j}$ wrt time. $T_{i,q}^{sink}(t)$ denotes temperature of $\varpi_{i,q}$ wrt time and is given by Eq. 6, where $\frac{dT_{i,q}^{sink}(t)}{dt}$ and T^{amb} are respectively the derivative of temperature of $\varpi_{i,q}$ wrt time and ambient temperature of the system.

$$C_i \frac{dT_{i,j}(t)}{dt} = \Phi_{i,j}(t) - \sum_{j'=1}^k \omega_{j,j'}^i (T_{i,j}(t) - T_{i,j'}(t)) - \sum_{q=1}^h \zeta_{j,q}^i (T_{i,j}(t) - T_{i,q}^{sink}(t)) \quad (5)$$

$$C_i^{sink} \frac{dT_{i,q}^{sink}(t)}{dt} = -\omega^{amb} (T_{i,q}^{sink}(t) - T^{amb}) - \sum_{j=1}^k \zeta_{j,q}^i (T_{i,q}^{sink}(t) - T_{i,j}(t)) - \sum_{q'=1}^h \omega_{q,q'}^i (T_{i,q}^{sink}(t) - T_{i,q'}^{sink}(t)) \quad (6)$$

As long as $M_{i,j}$ is running at a constant frequency, its temperature is a non-decreasing function [10], [11], [12], [24]. When the system reaches a steady state condition, the temperature of $M_{i,j}$ becomes steady. $T_{i,j}^*$ denotes the peak (or maximum) temperature of $M_{i,j}$ running at constant frequency $f_{i,j}$ $\left(\frac{d^2 T_{i,j}(t)}{dt^2} \Big|_{T_{i,j}(t)=T_{i,j}^*} > 0\right)$, and $T_{i,q}^{sink*}$ denotes the peak temperature of the $\varpi_{i,q}$ $\left(\frac{d^2 T_{i,q}^{sink}(t)}{dt^2} \Big|_{T_{i,q}^{sink}(t)=T_{i,q}^{sink*}} > 0\right)$. Eq. 7 approximates the values of $T_{i,j}^*$ and $T_{i,q}^{sink*}$.

$$0 = (\gamma_i f_{i,j} + \chi_i f_{i,j}^3) + T_{i,j}^* \left(\delta_i f_{i,j} - \sum_{j'=1}^k \omega_{j,j'}^i - \sum_{q=1}^h \zeta_{j,q}^i \right) + \sum_{j'=1}^k \omega_{j,j'}^i T_{i,j'}^* + \sum_{q=1}^h \zeta_{j,q}^i T_{i,q}^{sink*} \quad (7a)$$

$$0 = \omega^{amb} T^{amb} + T_{i,q}^{sink*} \left(-\omega^{amb} - \sum_{j=1}^k \zeta_{j,q}^i - \sum_{q'=1}^h \omega_{q,q'}^i \right) + \sum_{j=1}^k \zeta_{j,q}^i T_{i,j}^* + \sum_{q'=1}^h \omega_{q,q'}^i T_{i,q'}^{sink*} \quad (7b)$$

We get Eq. 8 from simplifying Eq. 7 [12], where

$$\begin{aligned}
A_{j,j} &= \delta_i f_{i,j} - \sum_{j'=1}^k \omega_{j,j'}^i - \sum_{q=1}^h \zeta_{j,q}^i \\
A_{j,j'} &= \omega_{j,j'}^i \\
A_{j,k+q} &= A_{k+q,j} = \zeta_{j,q}^i \\
A_{k+q,k+q} &= -\omega^{amb} - \sum_{j=1}^k \zeta_{j,q}^i - \sum_{q'=1}^h \omega_{q,q'}^i \\
A_{k+q,k+q'} &= \omega_{q,q'}^i
\end{aligned}$$

$$\begin{pmatrix} A_{1,1} & \dots & A_{1,k+h} \\ \vdots & \ddots & \vdots \\ A_{k,1} & \dots & A_{k,k+h} \\ A_{k+1,1} & \dots & A_{k+1,k+h} \\ \vdots & \ddots & \vdots \\ A_{k+h,1} & \dots & A_{k+h,k+h} \end{pmatrix} \begin{pmatrix} T_{i,1}^* \\ \vdots \\ T_{i,k}^* \\ T_{i,k+1}^{sink*} \\ \vdots \\ T_{i,k+h}^{sink*} \end{pmatrix} = - \begin{pmatrix} \gamma_i f_{i,1} + \chi_i f_{i,1}^3 \\ \vdots \\ \gamma_i f_{i,k} + \chi_i f_{i,k}^3 \\ \omega^{amb} T^{amb} \\ \vdots \\ \omega^{amb} T^{amb} \end{pmatrix} \quad (8a)$$

$$\text{Therefore, } [T]_{k+h,1} = -[A]_{k+h,k+h}^{-1} \times [\lambda]_{k+h,1} \quad (8b)$$

E. Problem Definition

Proposition 1: If the frequency of $M_{i,j}$ is at least $\tilde{U}_{i,j}$, then any periodic hard real-time scheduling policy which can fully utilize the core (e.g., Earliest Deadline First (EDF), Least Laxity First) can be used to obtain a feasible schedule [25].

According to Proposition 1, we assume that the frequency of $M_{i,j}$ is tightly related to the actual utilization $\tilde{U}_{i,j}$ instead of the worst-case utilization $U_{i,j}$. The actual frequency of $M_{i,j}$ denoted by $\tilde{f}_{i,j}$ is thus the lowest discrete frequency that is greater than or equal to $\tilde{U}_{i,j}$. The energy consumption of $M_{i,j}$ during the time interval $[0, P]$ is thus estimated by $P \times \Phi(\tilde{f}_{i,j})$. We also need to make sure that by running $M_{i,j}$ at frequency $\tilde{f}_{i,j}$, the peak temperature of $M_{i,j}$ denoted by $T_{i,j}^*$ is no more than $T_{i,j}^{max}$, which is the maximum feasible operating temperature of $M_{i,j}$.

1) **Problem Statement:** Given a set of periodic real-time tasks (Γ) and a set of interconnected multi-core multiprocessor units (Ω), the problem is to identify a subset of cores to be activated (Ψ) and partition tasks to the active cores such that the overall energy consumption is minimized, peak temperature constraints are satisfied, and task deadlines are not violated. This problem is known to be $\mathcal{NP}$ -Hard in the strong sense [4], [26]. Thus, the problem is formally defined as follows:

$$\text{Minimize: } P \sum_{M_{i,j} \in \Psi} \Phi_{i,j}(\tilde{f}_{i,j}) \quad \begin{matrix} i=1 \dots m \\ j=1 \dots k \\ \Psi \subseteq \Omega \end{matrix} \quad (9)$$

$$\text{Subject to: } 0 \leq \tilde{U}_{i,j} \leq \beta_{i,j} \quad \begin{matrix} i=1 \dots m \\ j=1 \dots k \\ M_{i,j} \in \Psi \end{matrix} \quad (10a)$$

$$\sum_{M_{i,j} \in \Psi} \alpha_i \tilde{U}_{i,j} = \gamma U_{tot} \quad \begin{matrix} 0 \leq l \leq m \\ 0 \leq c_i \leq k \\ \Psi \subseteq \Omega \end{matrix} \quad (10b)$$

$$\tilde{f}_{i,j} : \text{lowest discrete frequency where } \tilde{f}_{i,j} \geq \tilde{U}_{i,j} \quad \begin{matrix} i=1 \dots m \\ j=1 \dots k \\ M_{i,j} \in \Psi \end{matrix} \quad (10c)$$

$$T_{i,j}^* \leq T_{i,j}^{max} \quad \begin{matrix} i=1 \dots m \\ j=1 \dots k \\ M_{i,j} \in \Psi \end{matrix} \quad (10d)$$

where, $\beta_{i,j}$ is the schedulable utilization bound (e.g. 1.0 if $M_{i,j}$ is under EDF) and γ is the ratio between actual and worst-case total utilization of the tasks.

IV. ADAPTIVE PARTITIONING ALGORITHMS

In this section we present adaptive partitioning algorithms which are based on feedback control and real-time scheduling theories and use actual utilization of tasks to minimize the overall energy consumption.

During the initial partitioning of tasks, the values of the actual utilization of tasks are hard to estimate. Thus, for the initial partitioning it is assumed that actual and worst-case utilization of tasks are same, i.e. $\tilde{u}_i = \frac{W_i}{P_i}$ ($i = 1 \dots n$) and $\gamma = 1$. This information is used to compute the initial partitioning using the *Min-core Worst-fit* (MW) heuristic which is described below. Thus, we achieve the initial partitioning of tasks and also identify the set of active cores (Ψ) needed to execute all the tasks without violating any constraints.

In MW heuristic, we use worst-case execution time of tasks to partition the tasks into cores according to the *worst-fit* scheduling strategy. Therefore, the core with maximum available capacity is selected to execute the next task in the queue [4], [9]. But, worst-fit scheduling just solves the partitioning part of the problem. It does not find the least number of active cores (Ψ) needed to execute all the tasks. In order to find Ψ , the cores are sorted in decreasing order of their respective maximum capacity and the number of active cores is sequentially decreased when using worst-fit scheduling. Each partitioning scheme ensures that peak temperature constraints of cores and deadlines of tasks are not violated. Finally, the partitioning scheme resulting in the least energy consumption is identified as the initial partitioning strategy for the tasks. Thus, MW heuristic also identifies the set of active cores (Ψ) needed to execute all the tasks.

The initial partitioning of tasks and identification of active cores are based on the WCET of tasks which is generally known to be a pessimistic estimate and can be significantly greater than the actual execution time. Thus, we use a dynamic feedback based approach to adapt the initial partitioning using the actual utilization of tasks. We use a feedback predictor which monitors the current jobs and estimates the actual utilization of future jobs [20]. The information of $\tilde{u}_i$ is periodically updated and used to recalculate the partitioning of tasks for minimizing the energy consumption. However, unlike the initial partitioning, this new solution is not going to be implemented exactly since this dramatic change of task partitioning may require a large number of tasks migrations. Therefore, the new solution of the task partitioning is only used to determine the total utilization target for a core's tasks, denoted by $B_{j,r}$ ($j = 1 \dots m; r = 1 \dots k$), i.e. the target value of $\tilde{U}_{j,r}$. We set $B_{j,r} = \sum_{\tau_i \in \Gamma_{j,r}} \tilde{u}_i$, where $\Gamma_{j,r}$ denotes the tasks allocated to $M_{j,r}$ when the new partitioning scheme was to be followed. Since only a subset of active cores (Ψ) are needed to be activated, we have $B_{j,r} = \{0 | M_{j,r} \notin \Psi\}$. Thus, we develop utilization control methods to adapt the initial task partitioning and achieve the target utilization of the cores.

We use an approach similar to the controlled task migration approach [13] for changing the utilization of cores. Each job of a task is executed only on a single core while different jobs of the task can execute on different cores. Thus, the runtime context of a job is maintained in only one core but the task-level context may be migrated [13]. The task migration decisions of a core is done by the *utilization controller* of the core. The control period t_s is selected such that multiple jobs of a task can be released during the period. Therefore, $\tilde{U}_{j,r}(s)$ is estimated during the time interval $[(s-1)t_s, st_s]$ and the main objective of the task migration strategy is to choose migrations such that $\tilde{U}_{j,r}(s+1)$ is close to $B_{j,r}$ during the s^{th} sampling point (time st_s). Below we present two different task migration heuristics for accomplishing our objective.

Distributed Utilization Control (DUC) Heuristic: In DUC heuristic, a task is migrated to an under-utilized core ($\tilde{U}_{j,r} < B_{j,r}$) from an over-utilized core based on the migration probability. Let, $\mathcal{P}_{j,r,j',r'}$ denotes the probability of core $M_{j,r}$ migrating a task to core $M_{j',r'}$ ($j' = 1 \dots m; r' = 1 \dots k$) and $\delta_{j,r}(s)$ denotes the total actual utilization of tasks migrated out of $M_{j,r}$ during the $(s+1)^{th}$ control period. Therefore, the load dynamics of a core is given by Eq. 11.

$$\tilde{U}_{j,r}(s+1) = \tilde{U}_{j,r}(s) - \delta_{j,r}(s) + \sum_{j'=1}^m \sum_{r'=1}^k \frac{\alpha_{j',r'}}{\alpha_{j,r}} \mathcal{P}_{j',r',j,r} \delta_{j',r'}(s) \quad (11)$$

The current utilization of all under-utilized cores is broadcasted to all active cores. Based on this information, a core $M_{j,r}$ computes the values of $\mathcal{P}_{j,r,j',r'}$ using Eq. 12, where Θ denotes the set of under-utilized cores.

$$\mathcal{P}_{j,r,j',r'} = \begin{cases} 0 & \text{If } M_{j',r'} \notin \Theta \\ \frac{B_{j',r'} - \tilde{U}_{j',r'}(s)}{\sum_{M_{a,b} \in \Theta} (B_{a,b} - \tilde{U}_{a,b}(s))} & \text{Otherwise} \end{cases} \quad (12)$$

The control rules are given in Eq. 13. The value of positive control gain $K_{j,r}$ is selected to ensure overall system stability [27]. The *admission controller* in a core accepts a task migration if and only if the utilization and the task deadline miss ratios of the core are bounded (e.g. $\tilde{U}_{j,r}(s+1) \leq \beta_{j,r}$ and $MR_{j,r}(s+1) \leq 1\%$, $MR_{j,r}$ denotes the task deadline miss ratio of a core $M_{j,r}$). A small bound on miss ratio is selected to ensure soft real-time properties [28]. The admission controller also ensures that the peak temperature constraint of the core is not violated by accepting any new task. An idle core is turned off or put to a sleep state for saving energy.

$$y_{j,r}(s) = \tilde{U}_{j,r}(s) - B_{j,r} \quad (13a)$$

$$\delta_{j,r}(s) = \begin{cases} K_{j,r} y_{j,r}(s) & \text{If } y_{j,r}(s) \geq 0 \\ 0 & \text{Otherwise} \end{cases} \quad (13b)$$

Greedy Utilization Control (GUC) Heuristic: In GUC heuristic, we follow a greedy approach for migrating the tasks. The current utilization of all under-utilized cores is broadcasted to all active cores. Based on this information, the most over-utilized core selects the task with the least actual utilization and migrates the task to the most under-utilized core. The admission controller ensures that the utilization

and task deadline miss ratios of the core are bounded (e.g. $\tilde{U}_{j,r}(s+1) \leq \beta_{j,r}$ and $MR_{j,r}(s+1) \leq 1\%$) and the peak temperature constraint of a core is not violated by accepting a new task. An idle core is turned off or put to a sleep state for saving energy.

V. RESULTS AND DISCUSSION

In this section, we describe the simulation and analyze the results. Our results represent an average of 20 runs and all results have 95% confidence level. The maximum feasible operating temperature of a core and ambient temperature of the system are assumed to be 75°C and 0°C respectively. We used 6 different sets of periodic tasks, where the total number of tasks, n are 80, 90, 110, 140, 200, and 300. The total worst-case utilizations of task sets (U_{tot}) are assumed to be similar, and the average task utilizations (U_{tot}/n) of task sets are different. The values of the average task utilization are increasingly smaller with an average of 0.205, 0.175, 0.145, 0.115, 0.085, and 0.055 respectively. The actual task utilization is considered to be 40% of the worst-case utilization in our experiments. The task hyper-period P is 1000 seconds.

A. Results using Heat-Independent Thermal Model

We simulated a set of 8 interconnected multiprocessor units ($m = 8$), and each multiprocessor unit has 4 identical cores ($k = 4$). The parameters used in the simulation are given in Table I [11], [22], [24]. It is assumed that $f_i^{min} = 0.5 f_i^{max}$.

TABLE I
SIMULATION PARAMETERS USING HIT MODEL

M_i	f_i^{max}	γ_i	δ_i	χ_i	R_i	C_i	α_i
M_1	3.3	20.5060	0.1666	3.656	0.282	340	2.152
M_2	3.4	5.0187	0.1942	2.138	0.487	295	1.666
M_3	3.3	12.7880	0.2043	3.645	0.288	320	1.148
M_4	3.0	15.6262	0.1942	4.556	0.238	320	1.044
M_5	3.2	20.6393	0.1574	3.204	0.278	295	0.869
M_6	3.1	11.9759	0.1586	2.719	0.480	255	0.540
M_7	3.0	10.3490	0.1124	2.074	0.661	335	0.348
M_8	2.6	13.1568	0.1754	2.332	0.680	380	0.340

Fig. 1(a) gives the number of active cores ($|\Psi|$) needed to execute the respective task sets. Using WCET-based partitioning approach, 16 to 20 active cores are needed to execute all the task sets. However, using the actual utilization (AET) based DVS scaling, 7 or 8 active cores are sufficient.

For comparative analysis, we implemented MW heuristic using the actual task utilizations to get the baseline centralized solution. The tasks are migrated in the centralized solution for implementing the new task partitioning scheme, while tasks are migrated in the feedback approaches for achieving the utilization target (i.e. $B_{j,r}$, where $j=1, \dots, m; r = 1, \dots, k$). The task migrations are classified into two categories: *indispensable* and *normal* task migrations. The indispensable task migrations are composed of those unavoidable task migrations which results from turning off the cores. The normal task migrations represent the task migrations among the cores that remain active. Therefore, the number of normal task migrations are compared for evaluating the migration overhead of the heuristics.

Fig. 1. Results using Heat-Independent Thermal Model

The number of required normal task migrations is compared in Fig. 1(b). In our experiments, GUC and DUC heuristics reduce the number of normal task migrations on an average by 61.5% and 59.4% respectively as compared to the centralized solution. The number of indispensable tasks migrations in our experiments are 24, 32, 34, 39, 68, and 88 respectively when task set size are 80, 90, 110, 140, 200, and 300.

The total energy consumed by the different partitioning approaches is compared in Fig. 1(c). The energy consumption using the actual utilization (i.e. $\bar{U}_i$, $i=1, \dots, n$) based DVS scaling is significantly lower than the worst-case utilization (i.e. U_i) based DVS scaling. The energy consumption is further minimized by using adaptive or centralized strategies which use actual execution time of tasks for obtaining the partitioning of the tasks. In our experiments, the centralized solution reduces the energy consumption by 51.86% to 62.17% as compared to the WCET-based partitioning. DUC heuristic achieves similar reductions in energy consumption but requires 51.4% to 67.8% fewer normal task migrations as compared to the centralized solution. GUC heuristic also has similar energy savings but needs 58.5% to 65.8% fewer normal task migrations as compared to the centralized solution. Therefore, the proposed adaptive heuristics are effective in minimizing energy consumption according to the actual utilization of tasks and also minimizing the number of normal task migrations.

B. Results using Heat-Dependent Thermal Model

We simulated a set of 4 interconnected multiprocessor units ($m = 4$) where each multiprocessor unit has a 2×2 layout with 2 sinks, i.e. each unit has 4 identical cores ($k = 4$) [10], [12]. The simulation parameters are given in Table II [12] and we assume that $f_i^{min} = 0.5f_i^{max}$. The values of $A_{i,i}$ ($i = 1 \dots 4$) in matrix A can only be computed after the frequencies of the cores have been determined (Eq. 8). Therefore, these entries are left blank in Table II.

Fig. 2(a) compares the number of active cores ($|\Psi|$) needed to execute the respective task sets. 15 active cores are needed to execute all the task sets using WCET-based partitioning approach. However, 6 active cores are sufficient when actual utilization (AET) based DVS scaling is used.

The number of required normal task migrations is compared in Fig. 2(b). In our experiments, GUC and DUC heuristics reduce the number of normal task migrations on an average by 47.5% and 43.0% respectively as compared to the centralized

TABLE II
SIMULATION PARAMETERS USING HDT MODEL
(a) Matrix A for M_1

	0.009	0.004	0.000	0.200	0.050
0.009		0.000	0.004	0.050	0.060
0.004	0.000		0.009	0.200	0.050
0.000	0.004	0.009		0.050	0.060
0.200	0.050	0.200	0.050	-1.725	0.300
0.050	0.060	0.050	0.060	0.300	-1.445

(b) Matrix A for M_2

	0.025	0.007	0.020	0.500	0.050
0.025		0.020	0.007	0.050	0.200
0.007	0.020		0.025	0.500	0.050
0.020	0.007	0.025		0.050	0.200
0.500	0.050	0.500	0.050	-2.925	0.900
0.050	0.200	0.050	0.200	0.900	-2.325

(c) Matrix A for M_3

	0.020	0.010	0.000	0.400	0.100
0.020		0.000	0.010	0.100	0.120
0.010	0.000		0.020	0.400	0.100
0.000	0.010	0.020		0.100	0.120
0.400	0.100	0.400	0.100	-2.625	0.700
0.100	0.120	0.100	0.120	0.700	-2.065

(d) Matrix A for M_4

	0.013	0.007	0.004	0.300	0.050
0.013		0.004	0.007	0.080	0.090
0.007	0.004		0.013	0.300	0.080
0.004	0.007	0.013		0.080	0.090
0.300	0.080	0.300	0.080	-2.085	0.400
0.080	0.090	0.080	0.090	0.400	-1.665

(e) Other Simulation Parameters

M_i	f_i^{max}	γ_i	δ_i	χ_i	α_i
M_1	2.2	0.10	0.002	1.0	2.152
M_2	2.5	0.20	0.015	1.3	1.666
M_3	2.0	0.18	0.010	2.2	1.044
M_4	1.9	0.15	0.005	1.7	0.540

solution. Due to heat transfer among cores, HDT model requires a higher number of normal task migrations as compared to HIT model. The number of indispensable task migrations in our experiments are 23, 27, 31, 41, 62, and 89 respectively when the task set size are 80, 90, 110, 140, 200, and 300.

The total energy consumed by the partitioning schemes is compared in Fig. 2(c). The energy consumption using the actual utilization (i.e. $\bar{U}_i$) based DVS scaling is significantly smaller than the worst-case utilization (i.e. U_i) based DVS scaling. There is significant energy savings by using adaptive or centralized solutions that use actual execution time of tasks for obtaining the task partitioning. In our experiments, the centralized solution reduces the energy consumption by 58.5% to 62.2% as compared to the WCET-based solution.

Fig. 2. Results using Heat-Dependent Thermal Model

DUC heuristic achieves similar energy savings but requires 34.8% to 50% fewer normal task migrations as compared to the centralized solution. GUC heuristic also has similar energy savings and needs 39.5% to 50% fewer normal task migrations as compared to the centralized solution. Therefore, the proposed adaptive heuristics are effective in minimizing the number of normal task migrations and minimizing energy consumption according to the actual utilization of tasks.

VI. CONCLUSION

We present adaptive energy-efficient task partitioning for heterogeneous multi-core multiprocessor real-time systems. We use a power model which incorporates the impact of temperature and voltage of a core on its static power consumption. Two different thermal models, namely HIT and HDT models are used for estimating the peak temperature of a core. We present DUC and GUC heuristics for adaptive thermal-constrained energy-efficient partitioning of tasks which are feedback-based optimization and control approaches. In our simulations with HIT model, DUC and GUC heuristics minimize the energy consumption by an average of 55% as compared to a WCET-based task partitioning scheme and require an average of 60% fewer normal task migrations as compared to a centralized solution for obtaining similar energy savings. Similarly in our simulations with HDT model, DUC and GUC heuristics minimize the energy consumption by an average of 60% as compared to a WCET-based task partitioning scheme and require an average of 45% fewer normal task migrations as compared to a centralized solution for obtaining similar energy savings.

In future, we plan to investigate strategies for dealing with modeling inaccuracies of power and thermal parameters and task execution times. We also want to evaluate our solutions on several multiprocessor systems, e.g. multi-core computers, smart phones, and vehicle computing platforms.

REFERENCES

- [1] J.-J.Chen, A.Schranzhofer, and L.Thiele, "Energy minimization for periodic real-time tasks on heterogeneous processing units," *Parallel and Distributed Processing Symposium, International*, pp. 1–12, 2009.
- [2] R.Kumar and D. M.Tullsen, "Core architecture optimization for heterogeneous chip multiprocessors," in *PACT*, 2006, pp. 23–32.
- [3] Z.Wang and S.Ranka, "Thermal constrained workload distribution for maximizing throughput on multi-core processors," in *GREENCOMP*, 2010, pp. 291–298.
- [4] H.Aydin and Q.Yang, "Energy-aware partitioning for multiprocessor real-time systems," in *IPDPS*, 2003.
- [5] Y.Chen *et al.*, "Managing server energy and operational costs in hosting centers," *ACM SIGMETRICS PER*, vol. 33, no. 1, pp. 303–314, 2005.
- [6] I.Hong *et al.*, "Synthesis techniques for low-power hard real-time systems on variable voltage processors," in *RTSS*, 1998, pp. 178–187.
- [7] J.-J.Chen and C.-F.Kuo, "Energy-efficient scheduling for real-time systems on dynamic voltage scaling (dvs) platforms," in *RTCSA*, 2007.
- [8] P.DeLangen and B.Juurlink, "Leakage-aware multiprocessor scheduling for low power," in *IPDPS*, April 2006.
- [9] P.Langgen and B.Juurlink, "Leakage-aware multiprocessor scheduling," *J. Signal Process. Syst.*, vol. 57, pp. 73–88, October 2009.
- [10] T.Chantem, X. S.Hu, and R. P.Dick, "Temperature-aware scheduling and assignment for hard real-time applications on mpsoes," *IEEE Transactions on VLSI Systems*, no. 99, pp. 1 – 14, 2010.
- [11] G.Quan and V.Chaturvedi, "Feasibility analysis for temperature-constraint hard real-time periodic tasks," *Industrial Informatics, IEEE Transactions on*, vol. 6, no. 3, pp. 329–339, aug. 2010.
- [12] N.Fisher *et al.*, "Thermal-aware global real-time scheduling and analysis on multicore systems," in *Journal of System Architecture*, vol. 57, no. 5, 2011, pp. 547–560.
- [13] J.Carpenter *et al.*, "A categorization of real-time multiprocessor scheduling problems and algorithms," in *Handbook on Scheduling Algorithms, Methods, and Models*. Chapman Hall/CRC, Boca, 2004.
- [14] S. K.Baruah *et al.*, "Proportionate progress: A notion of fairness in resource allocation," *Algorithmica*, vol. 15, pp. 600–625, 1994.
- [15] S.Baruah, "Task partitioning upon heterogeneous multiprocessor platforms," in *RTAS*, 2004, pp. 536–543.
- [16] M.Goraczko *et al.*, "Energy-optimal software partitioning in heterogeneous multiprocessor embedded systems," in *DAC*, 2008, pp. 191–196.
- [17] A.Schranzhofer, J.-J.Chen, and L.Thiele, "Dynamic power-aware mapping of applications onto heterogeneous mpsoes platforms," *Industrial Informatics, IEEE Transactions on*, vol. 6, no. 4, nov. 2010.
- [18] Y.Liu *et al.*, "Accurate temperature-dependent integrated circuit leakage power estimation is easy," in *DATE*, 2007, pp. 1526–1531.
- [19] L.Schor *et al.*, "Worst-case temperature guarantees for real-time applications on multi-core systems," in *Proc. IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2012.
- [20] A.Block *et al.*, "An adaptive framework for multiprocessor real-time systems," *ECRTS*, pp. 23–33, 2008.
- [21] J.-J.Chen *et al.*, "Approximation algorithms for multiprocessor energy-efficient scheduling of periodic real-time tasks with uncertain task execution time," *RTAS*, pp. 13–23, 2008.
- [22] L.Wang and Y.Lu, "An efficient threshold-based power management mechanism for heterogeneous soft real-time clusters," *Industrial Informatics, IEEE Transactions on*, vol. 6, no. 3, pp. 352 –364, aug. 2010.
- [23] J. W. S. W.Liu, *Real-Time Systems*. Prentice Hall PTR, 2000.
- [24] V.Chaturvedi, H.Huang, and G.Quan, "Leakage aware scheduling on maximum temperature minimization for periodic hard real-time systems," in *ICIT*, 2010, pp. 1802–1809.
- [25] H.Aydin *et al.*, "Dynamic and aggressive scheduling techniques for power-aware real-time systems," in *RTSS*, 2001, pp. 95–105.
- [26] J. A.Stankovic *et al.*, "Implications of classical scheduling results for real-time systems," *Computer*, vol. 28, pp. 16–25, 1995.
- [27] J.Chiaasson *et al.*, "The effect of time delays on the stability of load balancing algorithms for parallel computations," *IEEE Trans. on Control Systems Technology*, pp. 932–942, 2005.
- [28] C.Lu *et al.*, "Performance specifications and metrics for adaptive real-time systems," *RTAS*, 2000.