Thermal-Constrained Energy-Aware Partitioning for Heterogeneous Multi-Core Multiprocessor Real-Time Systems

Shivashis Saha, Ying Lu, and Jitender S. Deogun
Department of Computer Science and Engineering,
University of Nebraska-Lincoln, Lincoln, NE 68588-0115, U.S.A.
Email: {ssaha,ylu,deogun}@cse.unl.edu

***Abstract*—Next-generation multi-core multiprocessor real-time systems consume less energy at the cost of increased power-density. This increase in power-density results in high heat-density and may affect the reliability and performance of real-time systems. Thus, incorporating maximum temperature constraints in scheduling of real-time task sets is an important challenge. This paper investigates thermal-constrained energy-aware partitioning of periodic real-time tasks in heterogeneous multi-core multiprocessor systems. We adopt a power model which considers the impact of temperature and voltage on a processor's static power consumption. Two types of thermal models are used to respectively capture negligible and non-negligible amount of heat transfer among cores. We develop a novel genetic-algorithm based approach to solve the heterogeneous multi-core multiprocessor partitioning problem. Extensive simulations were performed to validate the effectiveness of the approach. Experimental results show that integrating a worst-fit based partitioning heuristic with the genetic algorithm can significantly reduce the total energy consumption of a heterogeneous multi-core multiprocessor real-time system.**

I. Introduction

An increased awareness for conserving energy has resulted in tremendous research interests in energy-efficient, low-power design of computer systems [1]. Next generation multiprocessor real-time systems are becoming increasingly heterogeneous due to its relatively better performance and lower energy consumption compared to homogeneous counterparts [2]. However, energy efficiency of these recent multiprocessor systems comes at the cost of increased power-density. This leads to high heat-density which in turn adversely affects the reliability and performance of real-time systems [3].

A processor's power consumption is composed of static and dynamic power consumptions. The static power consumption is generated by the leakage current which is needed to maintain the activeness of the processor [1]. The dynamic power consumption dissipated from executing a task on the processor is a function of the processor's frequency [4]. This function is assumed to be a strictly convex and monotonically increasing function, which is usually represented by a polynomial of at least second degree [5]. *Dynamic voltage scaling* (DVS) techniques exploit the convex relationship to minimize the overall energy consumption [6]. Energy-aware scheduling strategies for homogeneous multiprocessor systems using the DVS techniques and assuming negligible leakage power consumption is well investigated [7]. However, the leakage current results in a significant static power consumption which is comparable to the dynamic power consumption [8]. Leakage-aware partitioning strategies for heterogeneous systems were investigated recently in [1], [9]. The increase in the temperature of a processor due to an increased heat-density has resulted in a recent research interest in temperature-aware multiprocessor scheduling to improve the reliability and performance of homogeneous real-time systems [10], [11], [12]. However, thermal-constrained energy-aware multiprocessor scheduling in heterogeneous real-time systems have not yet received much attention.

In this paper, we investigate thermal-constrained energy-aware partitioning-based scheduling 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. Given a set of periodic tasks and a heterogeneous multi-core multiprocessor real-time system, the problem is to identify cores to be activated, allocate tasks to cores, and determine the frequencies of these cores such that the overall energy consumption is minimized, maximum temperature constraints are satisfied, and deadlines of all tasks are ensured. We consider a power model which captures not only the leakage power consumption [9], but also the impact of temperature [12] and voltage [11] of a core on the leakage current. We use two types of thermal models: *heat-independent thermal (HIT) model* — negligible or no heat transfer among cores [11]; and *heat-dependent thermal (HDT) model* — non-negligible heat transfer among cores and sinks [10], [12]. We present a genetic-algorithm based approach to solve the heterogeneous multi-core multiprocessor partitioning problem. Extensive simulations are performed to validate the effectiveness of the approach. Experimental data shows that the proposed algorithm, HyWGA (*Hybrid Worst-fit Genetic Algorithm*), which integrates a Worst-fit based partitioning heuristic with the Genetic Algorithm, is most effective in generating an allocation strategy that satisfies all the constraints and results in the minimum energy consumption. The allocation strategy generated by HyWGA reduces the energy consumption by up to 11% and 21% for HIT and HDT models respectively, as compared to the MW (*Min-core Worst-fit*) heuristic.

II. Related Work

Extensive efforts have been made to study the multiprocessor real-time scheduling of periodic tasks [13]. In general, existing approaches can be categorized into two types: *global* and *partitioning-based* scheduling. In the global scheduling [14], all eligible tasks are assembled into a single queue, from which the global scheduler selects tasks for execution. On the contrary, the partitioning-based approach allocates each task to a single processor, and processors are scheduled independently [15]. Due to simplicity in design and implementation, task partitioning approaches are more practical than global scheduling approaches [16]. In this paper we focus on the partitioning approaches with the objective to make them energy-aware while satisfying thermal constraints.

Energy-aware scheduling strategies for homogeneous multiprocessor systems have been extensively investigated [7]. On the contrary, energy-aware scheduling strategies for heterogeneous multiprocessor systems have received a limited attention [1], where most of the work assumes a power model with a negligible leakage current [17]. The impact of non-negligible, fixed leakage current on energy-aware scheduling for heterogeneous systems was only investigated recently [1]. However, it has been proven that the leakage current of a processor changes super linearly with its temperature [18].

Temperature-aware scheduling of real-time systems has been investigated recently [10], [11], [12]. The authors of [12] and [10] respectively investigate scheduling sporadic and periodic tasks in homogeneous multiprocessor systems to minimize maximum temperatures. The feasibility checking problem for real-time periodic task sets under the maximum temperature constraint was studied in [11]. The thermal models proposed in [10] and [12] capture non-negligible heat transfer between different cores in a multi-core system. In these models, it was assumed that the leakage current is only impacted by the temperature of the core [10], [12], [18]. However, it has been proven that the leakage current is impacted not only by the temperature of a core, but also by its supply voltage [11].

To the best of our knowledge, the thermal-constrained energy-aware scheduling of periodic tasks in heterogeneous multi-core multiprocessor systems has not yet received much attention. In this paper, we investigate partitioning-based scheduling of periodic real-time tasks to minimize the total energy consumption, while satisfying maximum temperature constraints of heterogeneous multi-core multiprocessor systems. Our work is unique as we consider HIT [11] and HDT [10], [12] thermal models to estimate the temperature of a processor, and use a power model that captures the impact of both temperature [12] and voltage [11] on a processor's static power consumption.

III. System Models and Problem Definition

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

A. Multiprocessor Model

We consider a heterogeneous multiprocessor system, which is heterogeneous across multiprocessors, but homogeneous within a multiprocessor. That is, each multiprocessor may have different computational capacity, speeds/frequencies, and power and thermal parameters, but the cores within a given multiprocessor are homogeneous. Let $\Omega = \{\rho_1, \rho_2, \cdots, \rho_m\}$ be a set of interconnected heterogeneous multiprocessor units, where a unit ρ_i has k_i identical cores (or processors), i.e. $\rho_i = \{\rho_{i,1}, \rho_{i,2}, \cdots, \rho_{i,k_i}\}$ $(i = 1 \ldots m)$, which can support dynamic voltage scaling (DVS) and vary its frequency to one of the discrete levels in the range $[f_i^{min}, f_i^{max}]$. f_i^{max} (f_i^{min}) is the maximum (minimum) operating frequency of multiprocessor unit ρ_i. For simple representation, we normalize the core frequency with respect to f_i^{max}. A multiprocessor unit's throughput (or capacity) is assumed to be proportional to the normalized operating frequencies $f_{i,j}$ of its cores [19]. The capacity of ρ_i, denoted by μ_i, is thus expressed as $\sum_{j=1}^{k_i} \alpha_i f_{i,j}$, where α_i is the performance coefficient of ρ_i. In a heterogeneous multiprocessor system, higher values of α_i correspond to more powerful multiprocessor units. In the remainder of this paper, unless otherwise specified, frequency means normalized frequency.

B. Task Model

Let $\Gamma = \{\tau_1, \tau_2, \cdots, \tau_n\}$ be a set of independent periodic real-time tasks. A periodic task τ_q is an infinite number of task instances (jobs) released with periodicity P_q $(q = 1 \ldots n)$ [20]. The period P_q of a task τ_q also represents the relative deadline of the current job. The worst-case execution time of τ_q is W_q on a core of a standard multiprocessor $\wp$ with performance coefficient $\alpha_{\wp} = 1$ and is $\frac{W_q}{f}$ if the core has a constant frequency f. Thus, task τ_q's worst-case utilization under the maximum frequency of a standard core is $u_q = \frac{W_q}{P_q}$. Let U_{tot} denote the total utilization of the task set Γ under the maximum frequency of a standard core, i.e., $U_{tot} = \sum_{q=1}^{n} u_q = \sum_{q=1}^{n} \frac{W_q}{P_q}$. A *necessary* condition for a feasible schedule on the set Ω of multiprocessors is to have $U_{tot} \leq \sum_{i=1}^{m} k_i \alpha_i$, where k_i and α_i as defined in Section III-A denote the number of cores and the performance coefficient of multiprocessor unit ρ_i. We make this assumption throughout the paper. In a partitioning-based multiprocessor system, given a task-to-processor partitioning, we can calculate the worst-case utilization $U_{i,j}$ of a core $\rho_{i,j}$ under its maximum frequency, i.e. $U_{i,j} = \frac{\sum_{\tau_r \in \Gamma_{i,j}} W_r/P_r}{\alpha_i} = \frac{\sum_{\tau_r \in \Gamma_{i,j}} u_r}{\alpha_i}$, where $\Gamma_{i,j}$ represents the set of tasks being allocated to $\rho_{i,j}$. Since each task is allocated to exactly one core, we have $U_{tot} = \sum_{q=1}^{n} u_q = \sum_{i=1}^{m} \sum_{j=1}^{k_i} \alpha_i U_{i,j}$. We use P to denote the hyper-period of task set Γ, i.e., the minimum positive number P such that the released jobs are repeated every P time units. For instance, P is the least common multiple (LCM) of all task periods $P_1, \cdots, P_n$ when the periods are integral. Thus, our objective is to minimize the overall energy consumption in the hyper-period P while satisfying maximum temperature and real-time constraints.

C. Power Model

The power consumption of a core $\rho_{i,j}$ $(i = 1 \ldots m, j = 1 \ldots k_i)$, denoted by $\Phi_{i,j}$ is composed of two parts $\Phi_{i,j}^s$ and $\Phi_{i,j}^d$ (Eq. 1a). Here, $\Phi_{i,j}^s$ models the static (or leakage) power consumption generated by the leakage current required to maintain the activeness of the core [1], [8], and $\Phi_{i,j}^d$ models the dynamic power consumption dissipated from executing a task on the core [4]. The static power consumption of a core

$\rho_{i,j}$ denoted by $\Phi^s_{i,j}$ is closely approximated by Eq. 1b which models the dependence of leakage current on both the frequency $f_{i,j}$ [10], [12] (proportional to the supply voltage [21]) and the temperature $T_{i,j}$ [11], [21] of the core, where γ_i and δ_i are both non-negative constants dependent on the architecture of ρ_i. The dynamic power consumption of a core $\rho_{i,j}$ denoted by $\Phi^d_{i,j}$ is a function of its frequency, i.e. $\Phi^d_{i,j} = g(f_{i,j})$. In current DVS technologies, the function $g(f_{i,j})$ is assumed to be a strictly convex and monotonically increasing function, which is usually represented by a polynomial of at least second degree. As shown in Eq. 1c, we assume that $\Phi^d_{i,j}$ is a cubic polynomial in frequency, i.e. involves $f^3_{i,j}$ [10], [11], [12]. χ_i is a non-negative constant dependent on the architecture of ρ_i.

$$\Phi_{i,j}(f_{i,j}) = \Phi^s_{i,j}(f_{i,j}) + \Phi^d_{i,j}(f_{i,j}) \tag{1a}$$

$$\Phi^s_{i,j}(f_{i,j}) = \gamma_i f_{i,j} + \delta_i f_{i,j} T_{i,j} \tag{1b}$$

$$\Phi^d_{i,j}(f_{i,j}) = \chi_i f^3_{i,j} \tag{1c}$$

D. Temperature Model

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

1) Heat-Independent Thermal Model (HIT Model): We assume there is negligible or no heat transfer among cores of a multiprocessor unit and among different units [11], [21]. Using the RC thermal model [1], [10], [11], [12], [21], the temperature of a core with respect to (wrt) time, $T_{i,j}(t)$ ($i = 1 \dots m,\ j = 1 \dots k_i$), follows Eq. 2, where T^{amb}_i is the ambient temperature (in $^\circ C$), R_i is the thermal resistance (in $J/^\circ C$), and C_i is the thermal capacitance (in $Watt/^\circ C$) of ρ_i; $\Phi_{i,j}(t)$ is the power consumption of core $\rho_{i,j}$ wrt time (in $Watt$), and $\frac{dT_{i,j}(t)}{dt}$ is the derivative of $\rho_{i,j}$'s temperature wrt time.

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

Let the initial temperature of $\rho_{i,j}$ at time t_0 be $T^0_{i,j}$. If $\rho_{i,j}$ is running at a constant frequency $f_{i,j}$, then its final temperature at time t_1, denoted by $T^1_{i,j}$, is given by Eq. 3.

$$\frac{dT_{i,j}}{dt} = \frac{T^{amb}_i + \gamma_i R_i f_{i,j} + \delta_i R_i f_{i,j} T_{i,j} + \chi_i R_i f^3_{i,j}}{R_i C_i} - \frac{T_{i,j}}{R_i C_i} \tag{3a}$$

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

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

$$T^1_{i,j} = \frac{T^{amb}_i + \gamma_i R_i f_{i,j} + \chi_i R_i f^3_{i,j}}{1 - \delta_i R_i f_{i,j}} - \left(\frac{T^{amb}_i + \gamma_i R_i f_{i,j} + \chi_i R_i f^3_{i,j}}{1 - \delta_i R_i f_{i,j}} - T^0_{i,j} \right) e^{-\left(\frac{1-\delta_i R_i f_{i,j}}{R_i C_i}\right)(t_1 - t_0)} \tag{3d}$$

The temperature of $\rho_{i,j}$ is a non-decreasing function as long as it runs at a constant frequency (Eq. 3) [10], [11], [12], [21]. It will become steady when the system reaches a steady state condition $\left(\frac{dT_{i,j}(t)}{dt} = 0\right)$. The maximum temperature of the core running at constant frequency $f_{i,j}$ is denoted by $T^*_{i,j}$ $\left(\frac{d^2 T_{i,j}(t)}{dt^2}\Big|_{T_{i,j}(t) = T^*_{i,j}} > 0\right)$. $T^*_{i,j}$ is estimated using Eq. 4.

$$0 = \frac{T^{amb}_i + \gamma_i R_i f_{i,j} + \delta_i R_i f_{i,j} T^*_{i,j} + \chi_i R_i f^3_{i,j}}{R_i C_i} - \frac{T^*_{i,j}}{R_i C_i} \tag{4}$$

Simplifying Eq. 4, we get Eq. 5

$$T^*_{i,j} = \frac{\frac{T^{amb}_i + \gamma_i R_i f_{i,j} + \chi_i R_i f^3_{i,j}}{R_i C_i}}{\frac{1-\delta_i R_i f_{i,j}}{R_i C_i}} = \frac{T^{amb}_i + \gamma_i R_i f_{i,j} + \chi_i R_i f^3_{i,j}}{1 - \delta_i R_i f_{i,j}} \tag{5}$$

2) Heat-Dependent Thermal Model (HDT model): We assume there is non-negligible heat transfer among cores of a multiprocessor unit [10], [12]. We also assume that there is negligible or no heat transfer among different multiprocessor units. In a multiprocessor unit ρ_i ($i = 1 \dots m$), we assume there is a set of heat sinks, $\Xi_i = \{\varpi_{i,1}, \varpi_{i,2}, \cdots, \varpi_{i,h_i}\}$. These heat sinks are placed on top of the cores, and are only used for heat dissipation. These heat sinks do not generate any power. The transfer of heat among the cores and heat sinks of a multiprocessor unit can be closely approximated by modeling the dynamic heat transfer process using Fourier's Law, where each core acts as a discrete thermal element [10], [12].

Using RC thermal model [10], [12], we assume the thermal conductance between two cores $\rho_{i,j}$ and $\rho_{i,j'}$ is $\omega^i_{j,j'} (\forall j, j' \in 1 \dots k_i)$. Let $\zeta^i_{j,q}$ represent the vertical thermal conductance between core $\rho_{i,j}$ and sink $\varpi_{i,q}$ ($q = 1 \dots h_i$), and $\omega^i_{q,q'}$ be the horizontal thermal conductance between the heat sinks ($\forall q, q' \in 1 \dots h_i$). We assume that $\omega^i_{j,j'}$=$\omega^i_{j',j}$, $\omega^i_{j,j}$=0, $\omega^i_{q,q'}$=$\omega^i_{q',q}$, and $\omega^i_{q,q}$=0. The thermal conductance between the heat sink and the environment is denoted by ω^{amb}_i. The thermal capacitance of ρ_i and Ξ_i are denoted by C_i and C^{sink}_i respectively. The equation of $\rho_{i,j}$'s temperature wrt time, $T_{i,j}(t)$, is formulated as Eq. 6. In Eq. 6, $\frac{dT_{i,j}(t)}{dt}$ is the derivative of $\rho_{i,j}$'s temperature wrt time, and $\Phi_{i,j}(t)$ is $\rho_{i,j}$'s power consumption wrt time. Similarly, the equation of $\varpi_{i,q}$'s temperature wrt time, denoted by $T^{sink}_{i,q}(t)$, is formulated as Eq. 7. In Eq. 7, $\frac{dT^{sink}_{i,q}(t)}{dt}$ is the derivative of $\varpi_{i,q}$'s temperature wrt time, and T^{amb}_i is the ambient temperature.

$$C_i \frac{dT_{i,j}(t)}{dt} = \Phi_{i,j}(t) - \sum_{j'=1}^{k_i} \omega^i_{j,j'} (T_{i,j}(t) - T_{i,j'}(t)) - \sum_{q=1}^{h_i} \zeta^i_{j,q} (T_{i,j}(t) - T^{sink}_{i,q}(t)) \tag{6}$$

$$C^{sink}_i \frac{dT^{sink}_{i,q}(t)}{dt} = -\omega^{amb}_i (T^{sink}_{i,q}(t) - T^{amb}_i) - \sum_{j=1}^{k_i} \zeta^i_{j,q} (T^{sink}_{i,q}(t) - T_{i,j}(t)) - \sum_{q'=1}^{h_i} \omega^i_{q,q'} (T^{sink}_{i,q}(t) - T^{sink}_{i,q'}(t)) \tag{7}$$

The temperature of $\rho_{i,j}$ is a non-decreasing function as long as it runs at a constant frequency [10], [11], [12], [21]. It will become steady when the system reaches a steady state condition $\left(\frac{dT_{i,j}(t)}{dt}=0, \frac{dT^{sink}_{i,q}(t)}{dt}=0\right)$. The maximum temperature of the core running at constant frequency $f_{i,j}$ is denoted by $T^*_{i,j}$ $\left(\frac{d^2T_{i,j}(t)}{dt^2}|_{T_{i,j}(t)=T^*_{i,j}} > 0\right)$, and the maximum temperature of the heat sink $\varpi_{i,q}$ is denoted by $T^{sink*}_{i,q}$ $\left(\frac{d^2T^{sink}_{i,q}(t)}{dt^2}|_{T^{sink}_{i,q}(t)=T^{sink*}_{i,q}} > 0\right)$. The values of $T^*_{i,j}$ and $T^{sink*}_{i,q}$ are given by Eq. 8

$$
\begin{aligned}
0 &= (\gamma_i f_{i,j} + \delta_i f_{i,j} T^*_{i,j} + \chi_i f^3_{i,j}) - \sum_{j'=1}^{k_i} \omega^i_{j,j'}(T^*_{i,j} - T^*_{i,j'}) \\
&\quad - \sum_{q=1}^{h_i} \zeta^i_{j,q}(T^*_{i,j} - T^{sink*}_{i,q}) \\
&= (\gamma_i f_{i,j} + \chi_i f^3_{i,j}) + T^*_{i,j}\left(\delta_i f_{i,j} - \sum_{j'=1}^{k_i}\omega^i_{j,j'} - \sum_{q=1}^{h_i}\zeta^i_{j,q}\right) \\
&\quad + \sum_{j'=1}^{k_i}\omega^i_{j,j'}T^*_{i,j'} + \sum_{q=1}^{h_i}\zeta^i_{j,q}T^{sink*}_{i,q}
\end{aligned} \tag{8a}
$$

$$
\begin{aligned}
0 &= -\omega^{amb}_i(T^{sink*}_{i,q} - T^{amb}_i) - \sum_{j=1}^{k_i}\zeta^i_{j,q}(T^{sink*}_{i,q} - T^*_{i,j}) \\
&\quad - \sum_{q'=1}^{h_i}\omega^i_{q,q'}(T^{sink*}_{i,q} - T^{sink*}_{i,q'}) \\
&= \omega^{amb}_i T^{amb}_i + T^{sink*}_{i,q}\left(-\omega^{amb}_i - \sum_{j=1}^{k_i}\zeta^i_{j,q} - \sum_{q'=1}^{h_i}\omega^i_{q,q'}\right) \\
&\quad + \sum_{j=1}^{k_i}\zeta^i_{j,q}T^*_{i,j} + \sum_{q'=1}^{h_i}\omega^i_{q,q'}T^{sink*}_{i,q'}
\end{aligned} \tag{8b}
$$

Simplifying Eq. 8 we get Eq. 9 [12], where

$$
\begin{aligned}
A_{j,j} &= \delta_i f_{i,j} - \sum_{j'=1}^{k_i}\omega^i_{j,j'} - \sum_{q=1}^{h_i}\zeta^i_{j,q} \\
A_{j,j'} &= \omega^i_{j,j'} \\
A_{j,k_i+q} &= A_{k_i+q,j} = \zeta^i_{j,q} \\
A_{k_i+q,k_i+q} &= -\omega^{amb}_i - \sum_{j=1}^{k_i}\zeta^i_{j,q} - \sum_{q'=1}^{h_i}\omega^i_{q,q'} \\
A_{k_i+q,k_i+q'} &= \omega^i_{q,q'}
\end{aligned}
$$

$$
\begin{pmatrix}
A_{1,1} & \dots & A_{1,k_i+h_i} \\
\vdots & \vdots & \vdots \\
A_{k_i,1} & \dots & A_{k_i,k_i+h_i} \\
A_{k_i+1,1} & \dots & A_{k_i+1,k_i+h_i} \\
\vdots & \vdots & \vdots \\
A_{k_i+h_i,1} & \dots & A_{k_i+h_i,k_i+h_i}
\end{pmatrix}
\begin{pmatrix}
T^*_{i,1} \\ \vdots \\ T^*_{i,k_i} \\ T^{sink*}_{i,k_i+1} \\ \vdots \\ T^{sink*}_{i,k_i+h_i}
\end{pmatrix}
= -
\begin{pmatrix}
\gamma_i f_{i,1} + \chi_i f^3_{i,1} \\ \vdots \\ \gamma_i f_{i,k_i} + \chi_i f^3_{i,k_i} \\ \omega^{amb}_i T^{amb}_i \\ \vdots \\ \omega^{amb}_i T^{amb}_i
\end{pmatrix} \tag{9a}
$$

We denote Eq. 9a by $[A]_{k_i+h_i,k_i+h_i} \times [T]_{k_i+h_i,1} = -[\lambda]_{k_i+h_i,1}$ (9b)

$$\text{Thus,}\quad [T]_{k_i+h_i,1} = -[A]^{-1}_{k_i+h_i,k_i+h_i} \times [\lambda]_{k_i+h_i,1} \tag{9c}$$

E. Problem Definition

Consider a core of a multiprocessor unit, $\rho_{i,j}$, and a set of periodic real-time tasks allocated to the core whose utilizations sum up to U, satisfying $U \le \alpha_i$. That is, the set of tasks' utilization on core $\rho_{i,j}$ is $U_{i,j} = \frac{U}{\alpha_i} \le 1$. According to the work by Aydin et al. [22], if we make the frequency of a core to be at least $U_{i,j}$, any periodic hard real-time scheduling policy which can fully utilize the core (e.g., Earliest Deadline First, Least Laxity First) can be used to obtain a feasible schedule. We therefore assume that core $\rho_{i,j}$ $(i = 1\dots m, j = 1\dots k_i)$ chooses a frequency $\tilde{f}_{i,j}$, which is the lowest discrete frequency greater than or equal to $U_{i,j}$. Then, core $\rho_{i,j}$'s energy consumption $P \times \Phi_{i,j}(\tilde{f}_{i,j})$ in the interval $[0, P]$ can be estimated using Eq. 1. $T^{max}_{i,j}$ is the maximum allowed operating temperature of $\rho_{i,j}$. Since the system will eventually reach steady state with maximum temperature $T^*_{i,j}$, we need to make sure that setting $\rho_{i,j}$'s frequency at $\tilde{f}_{i,j}$, $T^*_{i,j}$ is no greater than $T^{max}_{i,j}$.

1) Problem Statement: Given a set of periodic real-time tasks (Γ), and a set of interconnected heterogeneous multi-core multiprocessor units (Ω), the problem is to identify the cores to be activated, allocate tasks to these active cores, and determine the frequencies of these cores such that the overall energy consumption is minimized, maximum temperature constraints are satisfied, and deadlines of all tasks are ensured. We assume that inactive cores can be turned off. This problem is known to be $\mathcal{NP}$-Hard in the strong sense [4], [23]. We state the problem as follows, where Ψ and c_i respectively represent the set of active multiprocessors and the number of active cores in ρ_i.

$$\textbf{Minimize:}\quad P\sum_{\rho_i\in\Psi}\sum_{j=1}^{c_i}\Phi_{i,j}(\tilde{f}_{i,j}) \quad 0 < c_i \le k_i \tag{10}$$

$$\textbf{Subject to:}\quad 0 \le U_{i,j} \le 1.0 \quad {}_{j=1\dots k_i}^{i=1\dots m} \tag{11a}$$

$$\sum_{\rho_i\in\Psi}\sum_{j=1}^{c_i}\alpha_i U_{i,j} = U_{tot} \quad 0 < c_i \le k_i \tag{11b}$$

$$\tilde{f}_{i,j} : \text{the lowest discrete frequency satisfying } \tilde{f}_{i,j} \ge U_{i,j} \quad {}_{j=1\dots c_i}^{\rho_i\in\Psi} \tag{11c}$$

$$T^*_{i,j} \le T^{max}_{i,j} \quad {}_{j=1\dots k_i}^{i=1\dots m} \tag{11d}$$

IV. Partitioning Algorithms

Below, we present three different partitioning algorithms.

A. Genetic Algorithm Based Partitioning Approach

Genetic algorithms are stochastic search techniques based on Darwin's "Theory of Evolution" [24]. In this section, we discuss our genetic algorithm based approach that solves the heterogeneous multi-core multiprocessor partitioning problem formed in Section III-E.

<2,1>	<1,1>	<2,3>	<1,1>	<1,2>	<2,1>	<2,2>

Fig. 1. Example of a chromosome

1) Initial Population: In a genetic algorithm, the initial population consists of a group of individuals (or chromosomes), where each of them represents a possible solution of the problem. A chromosome is composed of several genes that are usually represented by a random number, or a word or sentence over an alphabet, or a bit. We use "integer coding" [24] to represent a gene, i.e. each gene is represented by a pair of non-negative integers. Fig. 1 gives an example of a chromosome. In a chromosome, any gene q is represented as an integer pair $\langle i, j\rangle$, if the q^{th} ($q = 1 \dots n$) task (in Γ) is assigned to core $\rho_{i,j}$ ($i = 1 \dots m$, $j = 1 \dots k_i$) (in Ω). The number of genes in a chromosome is equal to n. In Fig. 1, the values of m, k_i, and n are chosen as $2, 3$, and 7 respectively.

2) Fitness Value: Each chromosome is assigned a *fitness value* which determines the effectiveness of the solution to the given problem. In our algorithm, the fitness value of a chromosome determines whether the corresponding task allocation satisfies the constraints in Eq. 11, and how effectively the task allocation minimizes the total energy consumption. Eq. 12 represents the fitness function of our genetic algorithm, which minimizes the overall energy consumption of the system while satisfying the maximum temperature constraints. If the allocation strategy corresponding to a chromosome violates Eq. 11, then the penalty of the chromosome is proportional to the frequency of the core which violates Eq. 11 (i.e. $\Lambda \times f_{i,j}$, where Λ is a large integer). Thus, a higher fitness value of a chromosome corresponds to an allocation strategy that results in a lower total energy consumption without violating the maximum temperature constraints.

$$Fitness = E_{max} - E_{chromo} \tag{12a}$$

$$E_{max} = P \sum_{i=1}^{m} \sum_{j=1}^{k_i} \Phi_{i,j}(f_{i,j} = f_{i,j}^{max}) \tag{12b}$$

$$E_{chromo} = P \sum_{i=1}^{m} \sum_{j=1}^{k_i} \Phi'_{i,j}(f_{i,j})$$

$$\text{where, } \Phi'_{i,j}(f_{i,j}) = \begin{cases} 0 & \text{If } \rho_{i,j} \text{ is inactive} \\ \Phi_{i,j}(f_{i,j}) & \text{If Eq. 11 is satisfied} \\ \Lambda \times f_{i,j} & \text{Otherwise} \end{cases} \tag{12c}$$

3) Crossover and Mutation: Crossover and Mutation are two genetic operators which are applied on the chromosomes to produce new offsprings, hopefully with higher fitness values [24]. *Top-mate* selection procedure is used to select chromosomes for *two-point crossover* [24], where the first chromosome for crossover is selected based on the order of fitness value and the second chromosome is selected randomly from all the chromosomes. In this type of crossover, the genes between the two randomly chosen points of the selected chromosomes are interchanged with each other. An example of a *two-point crossover* is given in Fig. 2.

Mutation is applied on the chromosomes to bring diversity in the population. The main objective of mutation is to avoid losing useful information in the process of evolution [24]. This also helps to improve the local search performance of the genetic algorithm. *Two-point mutation* is applied to a randomly selected chromosome. In this operation, the genes between the two randomly chosen points of the chromosome are assigned new values. Fig. 3 gives an example of *two-point mutation*.

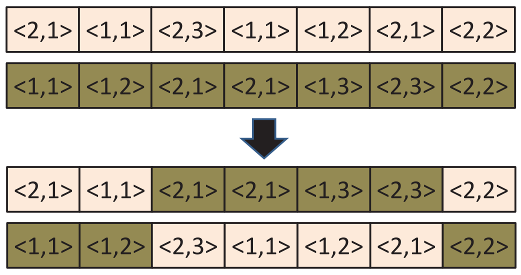


Fig. 2. Example of a two-point crossover

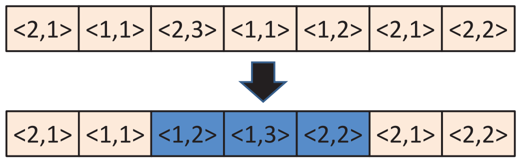


Fig. 3. Example of a two-point mutation

4) Elitism and Regeneration: To ensure that the best fitness value of the next generation is not worse than that of the current generation, a fixed number, MAX^{elite}, of best chromosomes are directly copied to the next generation. This operation is called *elitism* [24]. The remaining population of the next generation are generated using crossover and mutation operators.

5) Termination: The algorithm terminates when a fixed number, MAX^{gen}, of generations have been computed, or the algorithm converges, i.e. the best fitness value of the population has not changed for a fixed number of generations.

The pseudocode of the Genetic Algorithm (GA) based partitioning approach is given in Algorithm 1.

Algorithm 1 Genetic Algorithm Based Partitioning Approach

```
1: Randomly generate the initial population
2: generation ← 1
3: while generation ≤ MAX^gen do
4:     Compute fitness of the chromosomes using Eq. 12
5:     Rank the population based on the fitness value
6:     Copy MAX^elite best chromosomes to next generation, ties can be
       broken randomly
7:     Generate remaining population using crossover
8:     Mutate newly generated chromosomes
9:     if algorithm converged then
10:        break
11:    end if
12:    generation ← generation + 1
13: end while
14: print The total energy consumption, the allocation of tasks to cores, and
    the active/inactive states of cores that correspond to the chromosome with
    the best fitness value (Ties are broken randomly)
```

B. Min-Core Worst-Fit Partitioning Heuristic

In this partitioning heuristic, a task is allocated to a core according to the *worst-fit* scheduling strategy, i.e., the algorithm always chooses a core with the maximum remaining capacity to assign the next task [9], [4]. However, *worst-fit* scheduling only solves a part of the problem, i.e. it gives the allocation strategy. In order to find the optimal number of active cores, we need to explore applying the allocation strategy to a variable number of available cores [9], and thus identifying the optimal number of active cores needed to satisfy the constraints in Eq. 11. This motivates our Min-core Worst-fit partitioning (MW) heuristic described in Algorithm 2.

In Algorithm 2, the set of multiprocessors are sorted in decreasing order based on their respective performance coefficients. The algorithm sequentially decreases the number of cores used in the worst-fit scheduling strategy (Lines 1-3). The process of decrementing the number of available cores stops, when the set of real-time tasks cannot be successfully scheduled using the current number of active cores (Line 23). Lines 6 to 28 give the pseudocode for worst-fit scheduling strategy using a fixed number of cores (represented by the variable *used cores* in Line 30). The remaining capacities of all available cores are stored in a *heap* data-structure (Line 5). A new task is allocated to a core with the maximum remaining capacity that can successfully execute the task while satisfying the constraints in Eqs. 11a and d (Lines 7 to 10). It is possible that a core has a sufficient remaining capacity to execute a task, but its maximum temperature constraint is violated. In such a circumstance, the algorithm tries to identify the next core from the heap which can successfully accommodate the task while satisfying the temperature constraint (Lines 12 to 22). The total energy consumption of the system for a successful allocation strategy is computed in Line 29. The information about a successful allocation strategy is stored in a stack (Line 32), where each element of the stack represents a tuple of the allocation strategy, the number of cores used in the allocation strategy, and the total energy consumption of the allocation strategy (Line 31). Finally, the element in the stack corresponding to the allocation strategy which results in the minimum total energy consumption while satisfying the maximum temperature constraints is the output of the MW heuristic (Line 35 to 36). In case of a tie, the algorithm chooses an assignment strategy with the least number of active cores.

Algorithm 2 Min-Core Worst-Fit Partitioning Heuristic

```
Require: $\Omega = \{\rho_1, \rho_2, \cdots, \rho_m\}$, a set of interconnected heterogeneous
    multiprocessor units sorted in decreasing order based on their respective
    performance coefficient $\alpha_i (i = 1 \ldots m)$. $\Gamma = \{\tau_1, \tau_2, \cdots, \tau_n\}$, a set
    of independent periodic real-time tasks.
 1: for $i = m \to 1$ do
 2:     Number of available cores in $\rho_l$ is $k_l$, $(l = 1 \ldots i-1)$
 3:     for $j = k_i \to 1$ do
 4:         Number of available cores in $\rho_i$ is $j$
 5:         Construct a heap $\mathcal{H}$ of all available cores based on their remaining
            capacities, $\alpha_i \bar{U}_{i,j}$, where $\bar{U}_{i,j}$ is initialized to 1.0
 6:         for $q = 1 \to n$ do
 7:             if Allocating task $\tau_q$ to the core at the root of heap $\mathcal{H}$ satisfies
                Eqs. 11a and d then
 8:                 Allocate task $\tau_q$ to the core at the root of heap $\mathcal{H}$
 9:                 Change the core's remaining capacity $\alpha_i \bar{U}_{i,j}$ accordingly,
                    where $\bar{U}_{i,j}$ is updated as $\bar{U}_{i,j} = \bar{U}_{i,j} - \frac{u_q}{\alpha_i}$
10:                 Update $\mathcal{H}$
11:             else
12:                 Remove the root of $\mathcal{H}$
13:                 while $\mathcal{H} \neq$ empty do
14:                     if Allocating task $\tau_q$ to the core at the root of heap $\mathcal{H}$
                        satisfies Eqs. 11a and d then
15:                         Allocate task $\tau_q$ to the core at the root of heap $\mathcal{H}$
16:                         Change the core's remaining capacity $\alpha_i \bar{U}_{i,j}$ accord-
                            ingly, where $\bar{U}_{i,j}$ is updated as $\bar{U}_{i,j} = \bar{U}_{i,j} - \frac{u_q}{\alpha_i}$
17:                         Update $\mathcal{H}$
18:                         break
19:                     else
20:                         Remove the root of $\mathcal{H}$
21:                     end if
22:                 end while
23:                 if $\mathcal{H} =$ empty then
24:                     Insufficient number of available cores to schedule $\Gamma$
25:                     Goto Line 35
26:                 end if
27:             end if
28:         end for
29:         total energy $\leftarrow$ total energy consumption for this allocation strategy
            (using Eq. 1)
30:         used cores $\leftarrow \sum_{l=1}^{i} k_l + j$
31:         $\mathcal{E} \leftarrow$ (total energy, used cores, allocation strategy)
32:         push ($\mathcal{E}$) in stack $\mathcal{S}$
33:     end for
34: end for
35: $\mathcal{E} \leftarrow$ element in $\mathcal{S}$ with the minimum total energy consumption
36: print The total energy consumption, the allocation strategy, and the
    number of active cores corresponding to $\mathcal{E}$
```

C. Hybrid Worst-Fit Genetic Algorithm

To improve the performances of both GA approach (Alg. 1) and MW heuristic (Alg. 2), we present Hybrid Worst-fit Genetic Algorithm (HyWGA). HyWGA integrates the quickness of MW heuristic to generate a potential solution with the computational capability of GA based approaches. Thus, instead of randomly generating all the chromosomes of the initial population, HyWGA modifies the initial population by including a chromosome which represents a solution of the MW heuristic. The pseudocode of HyWGA is given in Algorithm 3.

A balanced allocation of tasks based on the available capacity of cores often does not result in the lowest overall energy consumption due to the heterogeneity of multiprocessors. However, such bin packing techniques combined with the optimal number of active cores (similar to MW heuristic) will result in potential solutions for the energy-aware partitioning problem. Such potential solutions can also be included in the initial population of the GA approach for improving its performance. Investigation of such strategies is a future work.

Algorithm 3 Hybrid Worst-Fit Genetic Algorithm

```
1: Encode a chromosome which corresponds to the solution generated by
   Algorithm 2
2: Randomly generate the rest of the initial population
3: Call Algorithm 1 on this initial population
```

V. Results and Discussion

In this section, we present the simulation and analyze the simulation results to compare the performance of the algorithms under both HIT and HDT thermal models. The maximum allowed operating temperature of a core and the ambient temperature of the system are assumed to be $65°C$ and

$0°C$ respectively. The task hyper-period P is 1000 seconds. The crossover and mutation rates for genetic algorithms are selected as 85% and 0.5% respectively. The value of MAX^{elite} is chosen as 1% of the population size.

A. Results using Heat-Independent Thermal Model

We assume a set of 8 interconnected heterogeneous multiprocessor units (i.e. $m = 8$), and each multiprocessor unit is assumed to have 8 identical cores ($k_i = 8, i = 1 \dots m$). Twelve different periodic real-time task sets are generated by changing the values of U_{tot} and the number of tasks (n). The task sets for a given value of U_{tot} were generated using normal distribution while selecting n as $100, 150, 200$, and 300 respectively. The values of U_{tot} were chosen as $20, 35$, and 45 respectively. The size of the initial population, and the maximum number of generations for the genetic algorithms are selected as 2000, and 10,000 respectively. The parameters used in the simulation are given in Table I[1] [11], [19], [21].

TABLE I
SIMULATION PARAMETERS USING HIT MODEL

ρ_i	f_i^{max}	γ_i	δ_i	χ_i	R_i	C_i	α_i
ρ_1	3.3	20.5060	0.1666	3.656	0.282	340	2.152
ρ_2	3.4	5.0187	0.1942	2.138	0.487	295	1.666
ρ_3	3.3	12.7880	0.2043	3.645	0.288	320	1.148
ρ_4	3.0	15.6262	0.1942	4.556	0.238	320	1.044
ρ_5	3.2	20.6393	0.1574	3.204	0.278	295	0.869
ρ_6	3.1	11.9759	0.1586	2.719	0.480	255	0.540
ρ_7	3.0	10.3490	0.1124	2.074	0.661	335	0.348
ρ_8	2.6	13.1568	0.1754	2.332	0.680	380	0.340

Traditional power models assume fixed static power consumption of a core [1]. For comparison purpose, we modified the MW heuristic to use the *traditional power model*, and denote it as M-MW heuristic. The estimated temperature and power of a core using the M-MW heuristic may not be accurate due to its ineffectiveness in incorporating the impact of both temperature and voltage on the leakage current. Thus, the actual temperature and power of a core may be significantly higher than its estimated value, which may result in violation of maximum temperature constraints and very high energy consumption, as observed in our simulation (Figs. 4(a)-4(c)).

The performance of the algorithms using the HIT model are compared in Fig. 4. The total energy consumption of the algorithms under different task sets are shown in Figs. 4(a)-4(c). Overall, HyWGA is most effective in generating an allocation strategy which results in the minimum energy consumption. In our experiments, HyWGA reduces the total energy consumption by 5% to 11% as compared to the MW heuristic. The total number of active cores used by the algorithms in their respective minimum-energy allocation strategies are compared in Figs. 4(d)-4(f). In Figs. 4(e) and 4(f), the minimum-energy allocation strategies generated by HyWGA use higher numbers of active cores than those generated by MW heuristic. The MW heuristic explores *worst-fit* scheduling strategy using a variable number of active cores. However, MW heuristic needs a set of interconnected heterogeneous multiprocessor units which is sorted in descending order based on their respective performance coefficients. Thus, MW heuristic is biased in favor of a low number of cores with high performance coefficients. This kind of selection strategy is more likely to identify a core with lower performance coefficient as an inactive core compared to a high performance coefficient core. This feature of MW heuristic prohibits it to explore an allocation strategy which can combine a mixture of cores with different performance coefficients. By leveraging a genetic algorithm and its computational effectiveness, HyWGA addresses this drawback, where it explores different task allocations without any biases in selection of the active cores. A comparison of the total energy consumption by using a different number of active cores in MW heuristic (when $U_{tot} = 20$ and $n = 150$) is shown in Fig. 4(h). It can be observed that there are several extreme points in Fig. 4(h). Thus, it is important to explore all the values of the number of active cores to identify the minimum-energy allocation strategy. In Fig. 4(h), using 10 or fewer cores in the allocation strategy cannot satisfy the constraints in Eq. 11.

The impact of sorting tasks (based on their utilizations) on the performance of the algorithms is shown in Fig. 4(g). If the algorithm uses a task set which has been sorted in decreasing order of task utilizations, then the name of the algorithm is appended with $-D$ (e.g. MW-D) in the legend of Fig. 4(g). Similarly, we append $-A$ to the name of the algorithm in the legend of Fig. 4(g), if it uses a task set which has been sorted in ascending order of task utilizations (e.g. GA-A). The MW heuristic has the best performance when a task set is sorted in a decreasing utilization order. This observation verifies that the *worst-fit* scheduling strategy has the best performance for *bin-packing* problem, when the items are sorted with decreasing sizes [4]. Due to a similar reasoning, the performance of the MW heuristic is adversely affected by sorting tasks in ascending order of their utilizations. In our experiments, the total energy consumption of the MW heuristic decreases by 0.2% to 3%, if task sets are sorted with decreasing task utilizations. However, this improvement in performance due to ordering of task sets is negligible in GA and HyWGA approaches.

The maximum fitness value of each generation of the GA and HyWGA are compared in Fig. 4(i) (when $U_{tot} = 35$ and $n = 200$). The maximum fitness value of the GA approach converges quickly to its optimal value. On the other hand, the maximum fitness value of HyWGA approach increases slowly from generation to generation, because the initial population's fitness value is close to its optimal value. Interestingly, randomness of initial population of GA approach sometimes results in an allocation strategy with the lowest energy consumption ($U_{tot} = 45, n = 100$). This advantage adversely affects the performance of the GA algorithm when there are a high number of tasks with a low total utilization. In this case, the randomness in GA approach causes its allocation strategy to use a high number of active cores (Fig. 4(d)), resulting in a higher total energy consumption.

[1]It is assumed that $f_i^{min} = 0.5 f_i^{max}$.

B. Results using Heat-Dependent Thermal Model

We assume a set of 4 interconnected heterogeneous multiprocessor units (i.e. $m = 4$), and assume the layout of the multiprocessor is 2×2 with 2 sinks, i.e. each unit has 4 identical cores ($k_i = 4, i = 1 \ldots m$) [10], [12]. Six different periodic real-time task sets are generated by changing the values of U_{tot} and the number of tasks (n). The task sets for a given value of U_{tot} were generated using normal distribution while selecting n as 100 and 200 respectively. The values of U_{tot} were chosen as $5, 10$, and 15 respectively. The size of the initial population, and the maximum number of generations for the genetic algorithms are selected as 200, and 500 respectively. The parameters used in the simulation are given in Table II[2] [12]. According to Eq. 9, the value of $A_{i,i}$ $(i = 1 \ldots 4)$ in matrix A can only be computed after the frequencies of cores have been determined. These entries are left blank in Table II.

TABLE II
SIMULATION PARAMETERS USING HDT MODEL

(a) Matrix A for ρ_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 ρ_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 ρ_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 ρ_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

ρ_i	f_i^{max}	γ_i	δ_i	χ_i	α_i
ρ_1	2.2	0.10	0.002	1.0	2.152
ρ_2	2.5	0.20	0.015	1.3	1.666
ρ_3	2.0	0.18	0.010	2.2	1.044
ρ_4	1.9	0.15	0.005	1.7	0.540

The HDT model captures the impact of heat transfer among different cores (Section III-D2). However, this transfer of heat among different cores is not captured in the HIT model (Section III-D1). Thus, when the heat transfer among cores is non-negligible, it is expected that the HDT model would result in a more accurate estimation of the core's temperature leading to a better estimation of energy consumption of a core. Let us consider multiprocessor unit ρ_2 for scheduling a task set. Let the non-normalized frequencies of $\rho_{2,1}$, $\rho_{2,2}$, $\rho_{2,3}$, and $\rho_{2,4}$ be 2, 1.9, 2.1, and 1.7 Ghz respectively. Accordingly to the HIT model, $T^*_{2,1} = 53.9°C$, $T^*_{2,2} = 46.1°C$, $T^*_{2,3} = 62.7°C$, and $T^*_{2,4} = 36.7°C$. This is expected as a lower frequency in the HIT model results in a lower maximum temperature (Eq. 5). However, there is heat transfer among cores. Applying the HDT model, the actual maximum temperatures of the cores with the same frequency assignment are, $T^*_{2,1} = 50.6°C$, $T^*_{2,2} = 66.5°C$, $T^*_{2,3} = 53.5°C$, and $T^*_{2,4} = 56.8°C$. These maximum temperature values reflect the impact of heat transfer among cores. For example, even if $\rho_{2,4}$ runs at a lower frequency, it still has a high temperature due to thermal conductivity between the cores and sinks. A more critical problem is: this frequency assignment violates the maximum temperature constraint for $\rho_{2,2}$, which fails to be detected by the HIT model. This example demonstrates the importance of using an accurate model, i.e., the HDT model to capture the heat transfer among cores. An inaccurate model may lead to the violation of temperature constraints, resulting in reduced system reliability and performance in a long run. Moreover, this inaccuracy can also result in inaccurate estimation of total energy consumption of the system, as observed in our simulation.

The performance of the algorithms using HDT model are compared in Fig. 5. In Fig. 5, the name of the algorithm is appended with -HIT, when the temperature was estimated using HIT model. Interestingly, the estimated total energy consumption using HIT model is within 1.6% of the value estimated using HDT model. Therefore, in special scenarios when temperature constraints could be relaxed, there is a trade-off between the accuracy in the estimation of energy consumption vs. the speed of computation of the thermal model. The total energy consumption of the algorithms under different task sets are shown in Figs. 5(a)-5(c). Overall, HyWGA approach is the best in generating an allocation strategy which results in the minimum energy consumption. In our experiments, HyWGA reduces the total energy consumption by up to 21% as compared to the MW heuristic. The total number of active cores used by the algorithms in their respective minimum-energy allocation strategies are compared in Figs. 5(d)-5(f). Due to the randomness in the initial population of GA algorithm, the minimum-energy allocation strategy for a task set with $U_{tot} = 5$ or 10 uses a higher than optimal number of active cores, resulting in a higher total energy consumption.

The impact of sorting a task set (based on task utilizations) on the performance of the algorithms is shown in Fig. 5(g). The legends used in Fig. 5(g) have similar meanings to those used in Fig. 4(g). We observe that, due to the heat transfer among cores, the MW heuristic may not always have the best performance when tasks are sorted in decreasing order of their utilizations. The maximum fitness value of each generation of

[2]It is assumed that $f_i^{min} = 0.5 f_i^{max}$.

the GA and HyWGA approaches are compared in Fig. 5(i) (when $U_{tot} = 10$ and $n = 100$). A comparison of the total energy consumption by using a different number of active cores by MW heuristic (when $U_{tot} = 10$ and $n = 100$) is shown in Fig. 5(h). In both of these figures, we observe a trend similar to that in Figs. 4(h)-4(i).

VI. Conclusion

This paper develops a genetic-algorithm based approach (HyWGA) to solve the thermal-constrained energy-aware partitioning problem for heterogeneous multi-core multiprocessor real-time systems. Our solution is novel in that it considers a power model that captures not only the leakage power consumption, but also the impact of temperature and voltage on the leakage current of a processor. We have used two different thermal models to estimate the temperature of a processor with both negligible and non-negligible amount of heat transfer among cores and sinks of a multiprocessor system.

We present several algorithms to identify cores to be activated and to allocate tasks to cores so as to minimize the total energy consumption while satisfying maximum temperature and deadline constraints. Extensive experimental simulations have been conducted to analyze the performance of the algorithms. From the analysis of experimental data, we observe that our HyWGA approach is most effective in minimizing the total energy consumption. It can minimize the total energy consumption by up to 11% and 21%, as compared to the MW heuristic, when using the HIT and HDT models respectively.

The off-line HyWGA approach developed in this paper is the first step towards solving the energy control problem for heterogeneous multi-core multiprocessor real-time systems. In the future, we will develop complementary and more efficient online strategies to handle practical issues like modeling inaccuracies of task execution times, power and thermal parameters, where task partitioning will be adapted according to the actual measurements of parameters.

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*, pp. 23–32, 2006.

[3] Z. Wang and S. Ranka, "Thermal constrained workload distribution for maximizing throughput on multi-core processors," in *GREENCOMP*, pp. 291–298, 2010.

[4] H. Aydin and Q. Yang, "Energy-aware partitioning for multiprocessor real-time systems," in *IPDPS*, 2003.

[5] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam, "Managing server energy and operational costs in hosting centers," *ACM SIGMETRICS PER*, vol. 33, no. 1, pp. 303–314, 2005.

[6] I. Hong, G. Qu, M. Potkonjak, and M. B. Srivastava, "Synthesis techniques for low-power hard real-time systems on variable voltage processors," in *RTSS*, pp. 178–187, 1998.

[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. de Langen and B. Juurlink, "Leakage-aware multiprocessor scheduling for low power," in *IPDPS*, April 2006.

[9] P. Langen 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 mpsocs," *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, pp. 329 –339, aug. 2010.

[12] N. Fisher, J.-J. Chen, S. Wang, and L. Thiele, "Thermal-aware global real-time scheduling on multicore systems," in *RTAS*, 2009.

[13] J. Carpenter, S. Funk, P. Holman, A. Srinivasan, J. Anderson, and S. Baruah, "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, N. K. Cohen, C. G. Plaxton, and D. Varvel, "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*, pp. 536–543, 2004.

[16] M. Goraczko, J. Liu, D. Lymberopoulos, S. Matic, B. Priyantha, and F. Zhao, "Energy-optimal software partitioning in heterogeneous multiprocessor embedded systems," in *DAC*, pp. 191–196, 2008.

[17] A. Schranzhofer, J.-J. Chen, and L. Thiele, "Dynamic power-aware mapping of applications onto heterogeneous mpsoc platforms," *Industrial Informatics, IEEE Transactions on*, vol. 6, nov. 2010.

[18] Y. Liu, R. P. Dick, L. Shang, and H. Yang, "Accurate temperature-dependent integrated circuit leakage power estimation is easy," in *DATE*, pp. 1526–1531, 2007.

[19] 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, pp. 352 –364, aug. 2010.

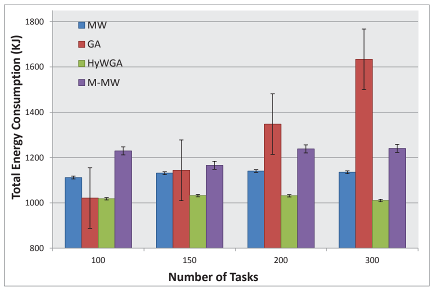
[20] J. W. S. W. Liu, *Real-Time Systems*. Prentice Hall PTR, 2000.

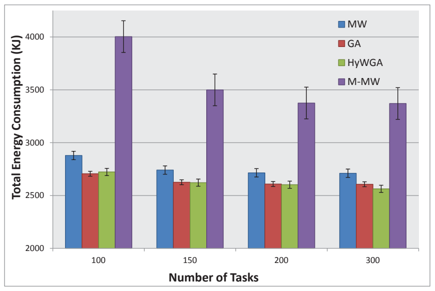
[21] V. Chaturvedi, H. Huang, and G. Quan, "Leakage aware scheduling on maximum temperature minimization for periodic hard real-time systems," in *ICCIT*, pp. 1802–1809, 2010.

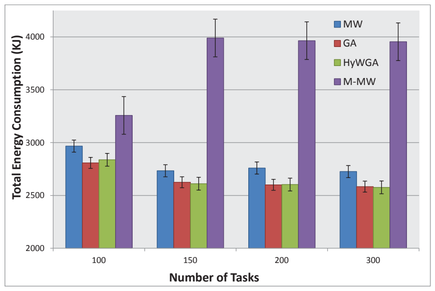
[22] H. Aydin, R. Melhem, D. Moss, and P. Mejla-Alvarez, "Dynamic and aggressive scheduling techniques for power-aware real-time systems," in *RTSS*, pp. 95–105, 2001.

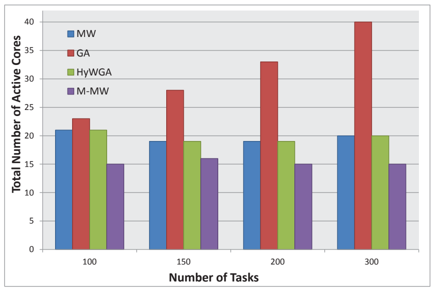
[23] J. A. Stankovic, M. Spuri, M. D. Natale, and G. C. Buttazzo, "Implications of classical scheduling results for real-time systems," *Computer*, vol. 28, pp. 16–25, 1995.

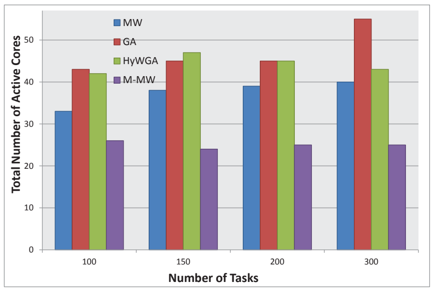
[24] D. E. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning*. Addison Wesley, 1989.

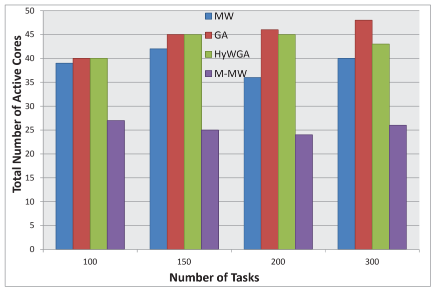

(a) Total Energy Consumption for $U_{tot} = 20$

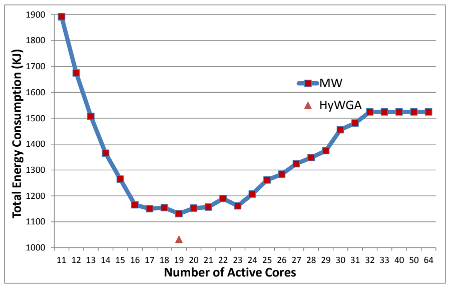

(b) Total Energy Consumption for $U_{tot} = 35$

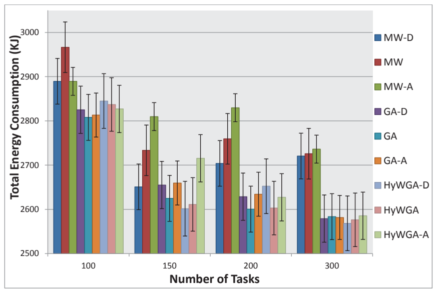

(c) Total Energy Consumption for $U_{tot} = 45$


(d) Total Number of Active cores for $U_{tot} = 20$


(e) Total Number of Active cores for $U_{tot} = 35$


(f) Total Number of Active cores for $U_{tot} = 45$




(g) Impact of ordering the tasks ($U_{tot} = 45$)

(h) Active Cores vs. Total Energy Consumption

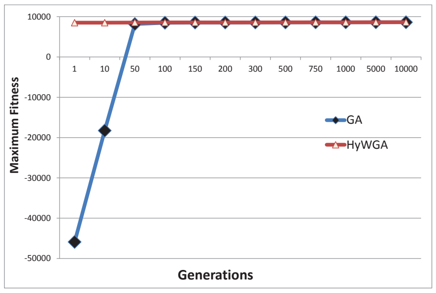
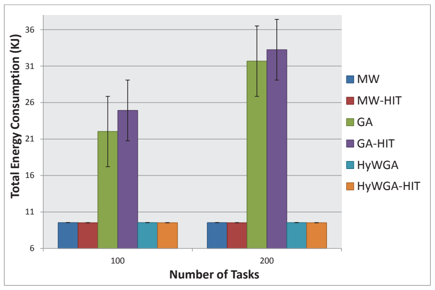
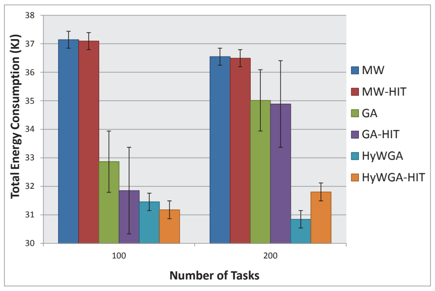

(i) Maximum Fitness vs. Generations

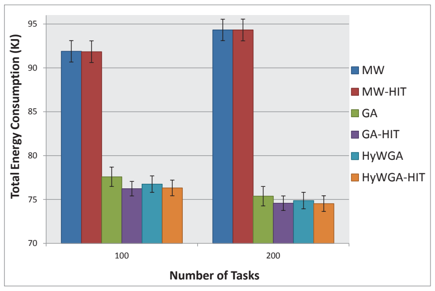
Fig. 4. Results using Heat-Independent Thermal Model



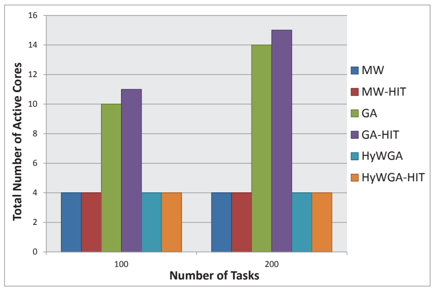
(a) Total Energy Consumption for $U_{tot} = 5$



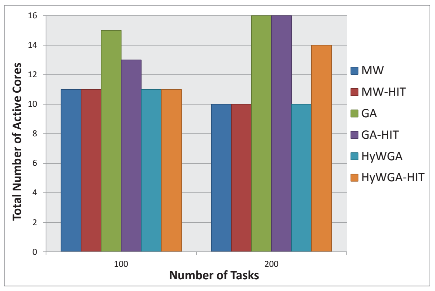
(b) Total Energy Consumption for $U_{tot} = 10$



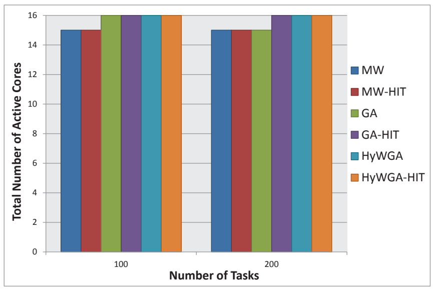
(c) Total Energy Consumption for $U_{tot} = 15$



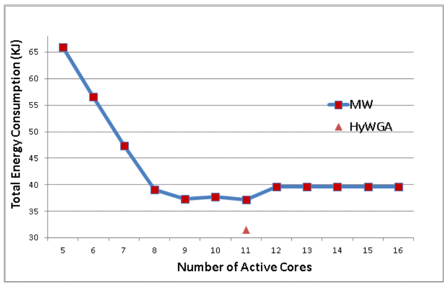
(d) Total Number of Active cores for $U_{tot} = 5$

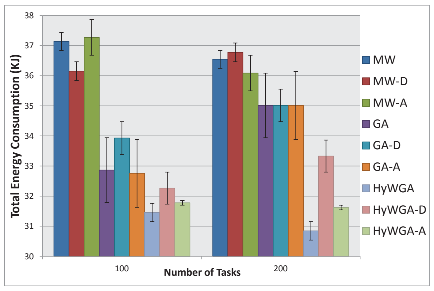


(e) Total Number of Active cores for $U_{tot} = 10$



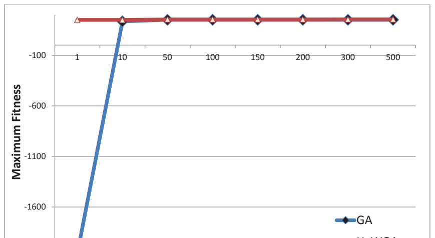
(f) Total Number of Active cores for $U_{tot} = 15$





(g) Impact of ordering the tasks ($U_{tot} = 10$)

(h) Active Cores vs. Total Energy Consumption


Maximum Fitness
GA