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

Björn Barrefors, 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: {bbarrefo,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. We use a simple thermal model with negligible amount of heat transfer among cores. We develop a novel approach to the heterogeneous multi-core multiprocessor partitioning problem by modeling it as the famous knapsack problem. Tests on a real cluster confirmed our power model. Experimental results show that integrating a branch-and-bound 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 a *heat-independent thermal model* with negligible or no heat transfer among cores [11]. We extend the work in [13] by proposing a new heuristic using a branch-and-bound-algorithm based approach to identify which cores to activate and come up with a new way of generating the initial generation to the genetic algorithm based approach to allocate tasks to cores (Hybrid Branch-and-Bound Genetic Algorithm; HyBaBGA). A real world testbed is used to confirm the adopted powermodel. Extensive simulations and experiements on a testbed cluster validate the effectiveness of our algorithm compared to earlier proposed algorithms MW (*Min-core Worst-fit*) and HyMWGA (*Hybrid Min-core Worst-Fit Genetic Algorithm*)[13]. The allocation strategy generated by HyBaBGA reduces the energy consumption by up to ??% and ??% , as compared to the MW (*Min-core Worst-fit*) heuristic and HyMWGA (*Hybrid Min-core Worst-Fit Genetic Algorithm*).

II. Related Work

Extensive efforts have been made to study the multiprocessor real-time scheduling of periodic tasks [14]. In general, existing approaches can be categorized into two types: *global* and *partitioning-based* scheduling. In the global scheduling [15], 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 [16]. Due to simplicity in design and implementation, task partitioning approaches are more practical than global scheduling approaches [17]. 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 [18]. 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 [19].

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], [19]. 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].

A recent paper [13] proposed a novel hybrid genetic algorithm for the thermal constrained heterogeneous multi-core multiprocessor scheduling problem using a power model which consider a power model which captures not only the impact of leakage current on the static power consumption, but also captures the impact of temperature and voltage of a processor on the leakage current. We not only improve on their work by using a testbed cluster to come up with an even more accurate power and thermal model and validates our results, but our work is also unique in its approach to the thermal constrained heterogeneous multi-core multiprocessor scheduling problem where we consider it as an instance of the well investigated $\mathcal{NP}$-Hard knapsack problem.

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 [20]. 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 $(q = 1 \ldots n)$ [21]. 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

In this paper we assume a general power model where 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]. 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. Equation 1b show the general power model used in our testbed, which was derived in [22]. This simplified model is only applicable for utilization above 99%, a more complex model is shown in [22] which includes utilization, due to the nature of the scheduling algorithms in this paper where frequency is adjusted to achieve a maximum utilization on each scheduled core we did not have to worry about this in our model.

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

$$\Phi_{i,j}(f_{i,j}) = a_{1_{i,j}} T^2_{i,j} + a_{2_{i,j}} f^2_{i,j} + a_{3_{i,j}} f_{i,j} T_{i,j} + a_{4_{i,j}} T_{i,j} + a_{5_{i,j}} f_{i,j} + a_{6_{i,j}} \tag{1b}$$

In our model we are interested in the power consumption at the steady state where a maximum temperature is reached. We ran a series of maximum utilization tasks on each core for all frequencies measuring the maximum temperatures and then applied a regression algorithm to find values for $a_{1_{i,j}}, a_{2_{i,j}}, a_{3_{i,j}}, a_{4_{i,j}}, a_{5_{i,j}}, a_{6_{i,j}}$ with r-squared values between 0.9959 and 0.9991.

D. Temperature Model

In this section, we will only focus on one thermal model, namely a *heat-independent thermal model* [11].

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], [23]. Using the RC thermal model [1], [10], [11], [12], [23], 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. 2a, 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.

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

$$T^{max}_{i,j}(t) - R_i \Phi_{i,j}(t) = T^{amb}_i \tag{2b}$$

Observe that at maximum temperature for $T_{i,j}(t)$, denoted $T^{max}_{i,j}(t)$, the change in temperature $\frac{dT_{i,j}(t)}{dt} = 0$ giving Eq. 2b. Our goal is to find an equation for the maximum temperature which can be calculated by the scheduler. By combining Eq. 2b and Eq. 1b we get 3c and can solve for $T^{max}_{i,j}(t)$ to find an equation dependant on known properties about the core. We here make the assumption that the ambient temperature of the cluster is known and static. To solve for $T^{max}_{i,j}(t)$ we are using the quadratic formula which is shown in Eq. 3a and 3b. We take Eq. 3c and put it in the same form as 3a to get Eq. 3d.

$$0 = A(T^{max}_{i,j}(t))^2 + B T^{max}_{i,j}(t) + C \tag{3a}$$

$$T^{max}_{i,j}(t) = \frac{-B + \sqrt{B^2 - 4AC}}{2A} \tag{3b}$$

$$T^{max}_{i,j}(t) = R_i(a_{1_{i,j}}(T^{max}_{i,j})^2 + a_{2_{i,j}} f^2_{i,j} + a_{3_{i,j}} f_{i,j} T^{max}_{i,j} - a_{4_{i,j}} T^{max}_{i,j} - a_{5_{i,j}} f_{i,j} + a_{6_{i,j}}) + T^{amb}_i \tag{3c}$$

$$0 = (R_i a_{1_{i,j}})(T^{max}_{i,j})^2 + (a_{3_{i,j}} f_{i,j} R_i + a_{4_{i,j}} - 1) T^{max}_{i,j} + (a_{2_{i,j}} f^2_{i,j} R_i + a_{5_{i,j}} f_{i,j} R_i + a_{6_{i,j}} R_i + T^{amb}_i) \tag{3d}$$

We can now use Eq. 3b with values for A, B, and C given in Eq. 4 as to calculate $T^{max}_{i,j}(t)$.

$$A = R_i a_{1_{i,j}} \tag{4a}$$

$$B = a_{3_{i,j}} f_{i,j} R_i + a_{4_{i,j}} - 1 \tag{4b}$$

$$C = a_{2_{i,j}} f^2_{i,j} R_i + a_{5_{i,j}} f_{i,j} R_i + a_{6_{i,j}} R_i + T^{amb}_i \tag{4c}$$

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 $U \leq \alpha_i$. That is, the set of tasks' utilization on core $\rho_{i,j}$ is $U_{i,j} = \frac{U}{\alpha_i} \leq 1$. According to the work by Aydin et al. [24], 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. 1b. $T^{max}_{i,j}$ 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], [25]. 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.

$$\textbf{Minimize:} \quad P \sum_{\rho_i \in \Psi} \sum_{j=1}^{c_i} \Phi_{i,j}(\tilde{f}_{i,j}) \quad 0 < c_i \leq k_i \tag{5}$$

$$\textbf{Subject to:} \quad 0 \leq U_{i,j} \leq 1.0 \quad {}^{i=1\dots m}_{j=1\dots k_i} \tag{6a}$$

$$\sum_{\rho_i \in \Psi} \sum_{j=1}^{c_i} \alpha_i U_{i,j} = U_{tot} \quad 0 < c_i \leq k_i \tag{6b}$$

$\tilde{f}_{i,j}$: the lowest discrete frequency satisfying $\tilde{f}_{i,j} \geq U_{i,j} \quad {}^{\rho_i \in \Psi}_{j=1\dots c_i}$ (6c)

$$T^*_{i,j} \leq T^{max}_{i,j} \quad {}^{i=1\dots m}_{j=1\dots k_i} \tag{6d}$$

IV. Optimal Subset Processor Selection

In this section we will only consider individual processors, as cores within a processor are homogeneous and to the best of our knowledge the possibility to turn off individual cores is not very common, it made more sense to only select processors. Our starting point was to investigate the performance of the HyWGA (*Hybrid Worst-fit Genetic Algorithm*)[13], described below, on an actual cluster. We found in our tests two results of great interest;

1) The genetic based approach schedules use a large number of processors [add ref to data].
2) HyWGA schedules tasks on the exact same processors as MW [add ref to data].

This led us to the conclusion that the genetic algorithm wasn't powerful enough to find the optimal processors but served the purpose of balance the tasks among the optimal processors. Genetic mutation is best applied to a population of similar chromosomes. We decided that to better use the genetic algorithm we should split the problem into two parts;

1) Find the optimal subset of processors for the schedule.
2) Balance tasks among selected subset of processors.

Before we define this problem we need to redefine the maximum frequncy of a processor i (f_i^{max}) as the maximum frequncy for which the processor satisfies the maximum temperature constraint using Eq. 3b and 4. Secondly we will assume monotonicity in power consumption with respect to frequency between two processors, expressed as; IF $\Phi_i(f_i = f_i^{max}) < \Phi_j(f_j = f_j^{max})$ THEN $\Phi_i(f_i = (f_i^{max} - f^k)) \leq \Phi_j(f_j = f_j^{max} - f^k)$ where $i! = j$ and f^k is an arbitrary constant frequency such that $(f_i^{max} - f^k) \geq f_i^{min}$ AND $(f_j^{max} - f^k) \geq f_j^{min}$. We will first introduce the proposed core selection algorithm in [13] (*Min-Core Worst-Fit*), and then propose an improved selection algorithm (*Branch-and-Bound*).

A. Min-Core Worst-Fit Based Selection

A worst-fit algorithm schedules tasks on the processor with maximum remaining capacity. Since we assume inactive processors can be turned off we would like to minimize the number of active processors. This can be achieved in iterations, each iteration the processor with lowest capacity is removed until no valid schedule exists. At this point the latest valid schedule is used. Processors with at least one task scheduled on it in the last valid schedule is selected as the optimal subset processors. Below is a modified algorithm to achieve the same result.

Algorithm 1 Min-Core Worst-Fit Selection

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 $\alpha_i (i = 1 \dots m)$. U_{tot}, total utilization of all processors. $\mathcal{S} = \varnothing$, the set of selected processors

1: **while** $\sum_{\rho_i \in \mathcal{S}} \alpha_i < U_{tot}$ **do**
2: Select the processor with greatest α and add to S
3: **end while**
4: **return** $\mathcal{S}$

B. Branch-and-Bound Based Selection

The MW selection algorithm is powerful in the sense that it is very fast and will find a good solution, as minimum number of processors is often a very good solution. However it is not a general solution. To always find the optimal solution we would need to do an exhaustive search on the set S of all subsets of all processors. The subset s with lowest power consumption for all processors i in the subset running on maximum frequency, see Eq. 8, which fulfilles the constraint in Eq. 7 is the optimal subset of processors. Using exhaustive search quickly becomes very costly as the number of processors increase, so we need to find a better way of finding the optimal subset. The key to finding a better solution is to notice that our problem is a constraint optimization problem, very similar to the famous knapsack problem where, given n items of known weights $w_1, \dots, w_n$ and values $v_1, \dots, v_n$ and a knapsack of capacity W, find the most valuable subset of the items that fit into the knapsack [26]. We will now show how the heterogeneous multiprocessor thermal constraint scheduling problem can be mapped to an instance of the knapsack problem. Remember that we already removed the thermal constraint by redefining maximum frequency above. Instead we define the necessery weight constraint as the total capacity of all selected processors, which need to be greater than or equal to the total utilization of all tasks, see Eq. 7 as defined in Section III-B. As value for each processor we use the power consumption at maximum frequency as defined in Eq. 1b, we are looking to minimize the total value.

$$U_{tot} \leq \sum_{i \in s} k_i \alpha_i \tag{7}$$

$$\sum_{i \in s} \Phi_i(f_i = f_i^{max}) \tag{8}$$

Branch-and-Bound is an algorithmic technique widely used for optimization problems such as the knapsack problem. The idea is to build a state-space tree where a branch is cut off as soon as we can conclude that the branch cannot contain the optimal solution [26]. We now go on to explain the algorithm. Processors are first ranked based on a payoff, at each state the processor with the greatest payoff is selected, one branch represents if the processor is selected, the other if it is rejected. For each branch a lower bound is calculated to estimate the capacity of the branch, the branch with lowest bound is then pursued. At each state we also need to check the constraint to make sure the branch can yield a valid schedule.

1) Initial State: In our algorithm we will have three different sets, Ω is the set of all processors, S is the set of selected processors and is initialized to $\varnothing$, and $\mathcal{R}$ is the set of discarded processors and is also initialized to $\varnothing$. We define $W = U_{tot}$ as the capacity constraint, at each state the current total weight w is the sum of capacities (Eq. 9) for all processors in $\Omega \setminus \mathcal{R}$ and value v is the sum of power comsumptions (Eq. 10) for all processors in S. w_i and v_i is the weight (capacity, Eq. 9) and value (power consumption, Eq. 10) of the processor with highest payoff (Eq. 11) in $\Omega \setminus (\mathcal{S} \cup \mathcal{R})$.

$$w_i = k_i \alpha_i \tag{9}$$

$$v_i = \Phi_i(f_i = f_i^{max}) \tag{10}$$

2) Payoff: Each processor is assigned a $payoff$ based on the ratio between its value and weight. In a minimization problem a lower $payoff$ is considered better. Unlike the common practice we will at each state select the processor least appropriate (greatest $payoff$) to try to discard bad candidates first instead of trying to select good candidates. This is due to how we check the constraint at each state. $payoff$ is defined in Eq. 11.

$$payoff_i = \frac{v_i}{w_i} \tag{11}$$

3) Lower bound: A lower bound is calculate for both branches of the current state to estimate future performance of that branch. The lower bound is calculated using the current value and the average payoff for remaining weight and is defined in Eq. 12. The branch with a lowest lowest bound is selected if it satisfies the constraint. We define $v_{(i+1}$ and w_{i+1} as the value (Eq. 10) and weight (Eq. 9) of the processor which would be considered at the next state, notice that this it is not the core currently under consideration.

$$lb = v + w(\frac{v_{i+1}}{w_{i+1}}) \tag{12}$$

The equation is terminated when $\Omega \setminus (\mathcal{S} \cup \mathcal{R}) == \varnothing$. At that point the set of selected processors S is returned. The pseudocode of the Branch-and-Bound (BaB) based selection algorithm is given in Algorithm 2.

Algorithm 2 Branch-and-Bound

Require: $\Omega = \{\rho_1, \rho_2, \cdots, \rho_m\}$, a set of interconnected heterogeneous multiprocessor units with payoffs $payoff_1 \geq payoff_1 \geq \ldots \geq payoffm$.

```
Initiate S and R to ∅
while Ω \ (S ∪ R) do
    Select processor ρ_i with highest payoff Eq. 11 from Ω \ (R ∪ S)
    Calculate lower bound using Eq. 12 for if processor ρ_i is selected (lb_w), and for if it is not (lb_wo)
    if lb_w < lb_wo or if Eq. 9 does not hold, where S in 9 is ⊗ \ (R ∪ {ρ_}) then
        Add the selected processor to S
    else
        Add the selected processor to R
    end if
end while
return S
```

V. Task Balancing Algorithm

In this section we describe how tasks are scheduled on the processors selected by any of the algorithms described above.

A. Genetic Based Algorithm

Genetic algorithms are stochastic search techniques based on Darwin's "Theory of Evolution" [27]. In this section, we discuss the genetic algorithm based approach that solves the second part of 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" [27] 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} $(q = 1 \ldots n)$ task (in Γ) is assigned to core $\rho_{i,j}$ $(i = 1 \ldots m,\ j = 1 \ldots k_i)$ (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. 6, and how effectively the task allocation minimizes the total energy consumption.

Eq. 13 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. 6, then the penalty of the chromosome is proportional to the frequency of the core which violates Eq. 6 (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.

$$Fitness = E_{max} - E_{chromo} \tag{13a}$$

$$E_{max} = P \sum_{i=1}^{m} \sum_{j=1}^{k_i} \Phi_{i,j}(f_{i,j} = f_{i,j}^{max}) \tag{13b}$$

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

$$\text{where, } \Phi_{i,j}^{'}(f_{i,j}) = \begin{cases} 0 & \text{If } \rho_{i,j} \text{ is inactive} \\ \Phi_{i,j}(f_{i,j}) & \text{If Eq. 6 is satisfied} \\ \Lambda \times f_{i,j} & \text{Otherwise} \end{cases} \tag{13c}$$

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 [27]. *Top-mate* selection procedure is used to select chromosomes for *two-point crossover* [27], 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

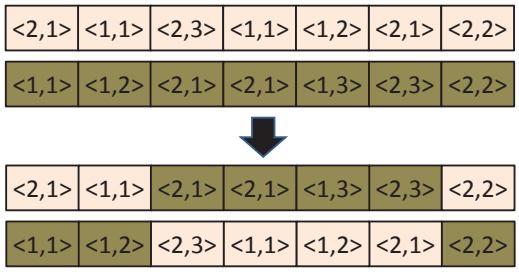

Fig. 2. Example of a two-point crossover

losing useful information in the process of evolution [27]. 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*.

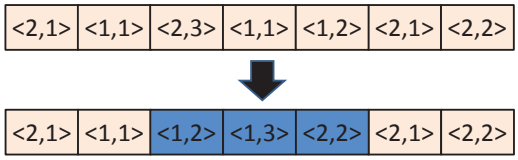

Fig. 3. Example of a two-point mutation

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* [27]. 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 3.

Algorithm 3 Genetic Algorithm Based Partitioning Approach

1: Randomly generate the initial population
2: generation $\leftarrow$ 1
3: **while** generation $\leq \text{MAX}^{gen}$ **do**
4: Compute fitness of the chromosomes using Eq. 13
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 $\leftarrow$ 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)

VI. Hybrid Processor Selection Genetic Scheduling Algorithm

To find a schedule we combine the above algorithms where first a subset of processors are selected using either 1 or 2, only the cores on these processors are available for scheduling tasks on. An initial population is randomly generated (limited to valid schedules). Feed the initial population into algorithm 3.

Algorithm 4 Hybrid Processor Selection Genetic Scheduling Algorithm

1: Get a set of processors from a selection algorithm, 1 or 2
2: Randomly generate a population of valid schedules on cores from selected processors
3: Call Algorithm 3 on this initial population

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] S. Saha, Y. Lu, and J. Deogun, "Thermal-constrained energy-aware partitioning for heterogeneous multi-core multiprocessor real-time systems," *Embedded and Real-Time Computing Systems and Applications (RTCSA), 2012 IEEE 18th International Conference on*, pp. 41–50, Aug 2012.
[14] 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.
[15] 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.
[16] S. Baruah, "Task partitioning upon heterogeneous multiprocessor platforms," in *RTAS*, pp. 536–543, 2004.
[17] 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.
[18] 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.
[19] 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.

[20] 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.
[21] J. W. S. W. Liu, *Real-Time Systems*. Prentice Hall PTR, 2000.
[22] S. Li, S. Wang, T. Abdelzaher, M. Kihl, and A. Robertsson, "Temperature aware power allocation: An optimization framework and case studies," *Sustainable Computing: Informatics and Systems*, vol. 2, no. 3, pp. 117–127, 2012.
[23] 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.
[24] 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.
[25] 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.
[26] A. V. Levitin, *Introduction to the Design and Analysis of Algorithms (2Nd Edition)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2006.
[27] D. E. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning*. Addison Wesley, 1989.