A Novel Algorithm for Thermal-Constrained Energy-Aware Partitioning in Heterogenous Multiprocessor Real-Time Systems

Björn Barrefors, Ying Lu, Shivashis Saha, and Jitender S. Deogun
Department of Computer Science and Engineering,
University of Nebraska-Lincoln, Lincoln, NE 68588-0115, U.S.A.
Email: {bbarrefo,ylu,deogun}@cse.unl.edu, and ssaha@huskers.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 power model. 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. Each multiprocessor may have different computational capacity, speeds/frequencies, and power and thermal parameters. Let $\Omega = \{\rho_1, \rho_2, \cdots, \rho_m\}$ be a set of interconnected heterogeneous multiprocessor units, where a unit ρ_i 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 its normalized operating frequencies f_i [20]. The capacity of ρ_i, denoted by μ_i, is thus expressed as $\sum_{j=1}^{k_i} \alpha_i f_i$, 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 standard multiprocessor $\wp$ with performance coefficient $\alpha_\wp = 1$ and is $\frac{W_q}{f}$ if the multiprocessor has a constant frequency f. Thus, task τ_q's worst-case utilization under the maximum frequency of a standard multiprocessor 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 multiprocessor, 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 $U_{tot} \leq \sum_{i=1}^{m} \alpha_i$, where α_i as defined in Section III-A denote 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 of a multiprocessor ρ_i under its maximum frequency, i.e. $U_i = \frac{\sum_{\tau_r \in \Gamma_i} W_r/P_r}{\alpha_i} = \frac{\sum_{\tau_r \in \Gamma_i} u_r}{\alpha_i}$, where Γ_i represents the set of tasks being allocated to ρ_i. Since each task is allocated to exactly one multiprocessor, we have $U_{tot} = \sum_{q=1}^{n} u_q = \sum_{i=1}^{m} \alpha_i U_i$. 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 multiprocessor unit, ρ_i, and a set of periodic real-time tasks allocated to the multiprocessor whose utilizations sum up to U, satisfying $U \leq \alpha_i$. That is, the set of tasks' utilization on multiprocessor ρ_i is $U_i = \frac{U}{\alpha_i} \leq 1$. According to the work by Aydin et al. [24], if we make the frequency of a multiprocessor to be at least U_i, 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 multiprocessor ρ_i ($i = 1 \dots m$) chooses a frequency $\tilde{f}_i$, which is the lowest discrete frequency greater than or equal to U_i. Then, multiprocessor ρ_i's energy consumption $P \times \Phi_i(\tilde{f}_i)$ in the interval $[0, P]$ can be estimated using Eq. 1b. T^{max}_i is the maximum allowed operating temperature of ρ_i. Since the system will eventually reach steady state with maximum temperature T^*_i, we need to make sure that setting ρ_i's frequency at $\tilde{f}_i$, T^*_i is no greater than T^{max}_i. From here on we will define $\tilde{f}^{max}_i$ to be the maximum qorking frequency of multiprocessor ρ_i such that the maximum temperature constraint is satisfied.

1) Problem Statement: Given a set of periodic real-time tasks (Γ), and a set of interconnected heterogeneous multiprocessor units (Ω), the problem is to identify the multiprocessors to be activated, allocate tasks to these active multiprocessors, and determine the frequencies of these multiprocessors such that the overall energy consumption is minimized, maximum temperature constraints are satisfied, and deadlines of all tasks are ensured. We assume that inactive multiprocessors 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 represent the set of active multiprocessors in ρ_i.

$$\textbf{Minimize:} \quad P \sum_{\rho_i \in \Psi} \Phi_i(\tilde{f}_i) \tag{5}$$

$$\textbf{Subject to:} \quad 0 \leq U_i \leq 1.0 \quad i = 1 \dots m \tag{6a}$$

$$\sum_{\rho_i \in \Psi} \alpha_i U_i = U_{tot} \tag{6b}$$

$$\tilde{f}_i \text{ : the lowest discrete frequency satisfying } \tilde{f}_i \geq U_i \qquad \rho_i \in \Psi \tag{6c}$$

$$T^*_i \leq T^{max}_i \quad i = 1 \dots m \tag{6d}$$

IV. Scheduling Algorithms

In this section we present two algorithms for solving the problem introduced in section III-E. A novel branch-and-bound algorithm and a previously suggested hybrid min-core worst-fit genetic algorithm.

A. Branch-and-Bound Algorithm

Branch-and-bound is an algorithmic technique for solving combinatorial optimization problems. Branch-and-bound algorithms have the same worst-case complexity as simple brute-force exhaustive search, however real-life tests have shown significant speed up in the average case. We use this technique to solve the problem of finding an optimal set of active multiprocessors. The idea behind branch-and-bound algorithms is to branch for each multiprocessor, one branch for if the multiprocessor is selected and one for if it is not. Each mutliprocessor ρ_i have a value v_i (Eq. 7), weight w_i (Eq. 8), and payoff p_i (Eq. 9) associated to it.

$$v_i = \Phi_i(f_i = \tilde{f}_i^{max}) \tag{7}$$

$$w_i = \mu_i \tag{8}$$

$$p_i = \frac{v_i}{w_i} \tag{9}$$

For each new branch we calculate a lower bound lb (Eq. 11, Alg. 3) representing the lowest possible value for $\hat{z}$ (Eq. 12) on that branch. If the current branch satisfies the minimum weight constraint in Eq. 13 we go ahead and commit to the new branch in Alg. 4.

$$x = x_1 \dots x_i \dots x_n \text{ where } \begin{cases} x_i = 0 \text{ if } \rho_i \text{ not selected} \\ x_i = 1 \text{ if } \rho_i \text{ selected} \end{cases} \tag{10}$$

$$r = \min\{j : \sum_{k=i}^{j} w_k > \hat{c}\} \tag{11a}$$

$$lb_i = \sum_{k=i}^{r-1} v_k + \lfloor (\hat{c} - \sum_{k=i}^{r-1} w_k) \frac{v_r}{w_r} \rfloor \tag{11b}$$

$$\hat{z} = \sum_{i=1}^{n} x_i v_i \tag{12}$$

$$\hat{c} = \sum_{i=1}^{n} x_i w_i \tag{13a}$$

$$\hat{c} >= W_{min} \tag{13b}$$

Once we commit the branch we proceed to compare the lower bound and the current value of $\hat{z}$ with the best solution found so far z. If $z > \hat{z} + lb_i$ better than the previously best branch z while satisfying Eq. 13 we update z in Alg. 5. x is defined as a list of 0's and 1's where $x_i = 0$ means multiprocessor ρ_i is not selected and $x_i = 1$ means ρ_i is selected (Eq. 10). If branch can't contain a valid solution or the solution is not better than the previously best we need to backtrack (6). When $i = n$ we backtrack to make sure our solution is the best.

Algorithm 1 Branch-and-Bound Algorithm

Require:
Number of processors n
Minimum required utilization W_{min}
Processors sorted based on payoff, $p_1 \leq p_2 \leq \ldots \leq p_n$

Ensure:
Returns the total power consumption ($\hat{z}$) and a list of all processors ($\hat{x}$) as either 0 or 1, where 1 is selected and 0 is not selected.

1: **procedure** BAB-A(n, W_{min}, v, w)
2: **goto** Alg. 2
3: **end procedure**

Algorithm 2 Part 1. Initialize

4: $z \leftarrow 0$
5: $\hat{z} \leftarrow 0$
6: $\hat{i} \leftarrow 0$
7: $v_{n+1} \leftarrow \infty$
8: $\hat{c} \leftarrow W_{min}$
9: $w_{n+1} \leftarrow \infty$
10: $\hat{x}_1 \leftarrow 0, \ldots, \hat{x}_n \leftarrow 0$
11: **goto** Alg. 3

1) Task Scheduling: We use a simple scheduling algorithm to assign tasks on the multiprocessors selected in Alg. 1 described in 7. TODO - More work needed here!

V. Hybrid Min-Core Worst-Fit Genetic Algorithm

HyWGA is a novel hybrid min-core worst-fit genetic algorithm developed by Saha et. al [13].

Algorithm 3 Part 2. Compute lower bound

12: $sum \leftarrow 0$
13: **for** $k \leftarrow j, n+1$ **do**
14: $sum \leftarrow sum + w_k$
15: **if** $sum > \hat{c}$ **then**
16: $r \leftarrow k$
17: **break**
18: **end if**
19: **end for**
20: **if** $r = n+1$ & $\hat{c} - \sum_{k=j}^{r-1} w_k > 0$ **then**
21: **goto** Alg. 6
22: **end if**
23: $lb_j \leftarrow \sum_{k=j}^{r-1} v_k + \lfloor (\hat{c} - \sum_{k=j}^{r-1} w_k) * \frac{v_r}{w_r} \rfloor$
24: **if** $z = 0$ **then**
25: **goto** Alg. 4
26: **end if**
27: **if** $z \leq \hat{z} + u$ **then**
28: **goto** Alg. 6
29: **end if**
30: **goto** Alg. 4

Algorithm 4 Part 3. Forward step

while $w_j \leq \hat{c}$ **do**
  $\hat{c} \leftarrow \hat{c} - w_j$
  $\hat{z} \leftarrow \hat{z} + v_j$
  $\hat{x}_j \leftarrow 1$
  $j \leftarrow j + 1$
end while
if $j \leq n$ **then**
  $\hat{x}_j \leftarrow 1$
  $\hat{c} \leftarrow \hat{c} - U_j$
  $\hat{z} \leftarrow \hat{z} + v_j$
  $j \leftarrow j + 1$
end if
if $\hat{c} > 0$ **then**
  goto Alg. 6
end if
goto Alg. 5

Algorithm 5 Part 4. Update best solution

if $\hat{z} < z$**or**$z = 0$ **then**
  $z \leftarrow \hat{z}$
  for $k \leftarrow 1, n$ **do**
    $x_k \leftarrow \hat{x}_k$
  end for
end if
$j \leftarrow n$
if $\hat{x}_n = 1$ **then**
  $\hat{c} \leftarrow \hat{c} + w_j$
  $\hat{z} \leftarrow \hat{z} - v_j$
  $\hat{x}_j \leftarrow 0$
end if
goto Alg. 6

Algorithm 6 Part 5. Backtrack

$i \leftarrow -1$
for $k \leftarrow j - 1, 1$ **do**
  if $\hat{x}_k = 1$ **then**
    $i \leftarrow k$
    break
  end if
end for
if $i = -1$ **then**
  return $\hat{z}$, $\hat{x}$
end if
$\hat{c} \leftarrow \hat{c} + w_j$
$\hat{z} \leftarrow \hat{z} - v_j$
$\hat{x}_j \leftarrow 0$
$j \leftarrow i + 1$
goto Alg. 3

Algorithm 7 Task Allocation Algorithm

Require:
  A set of real-time periodic tasks $\Gamma = \{\tau_1, \tau_2, \cdots, \tau_n\}$
  A set of multiprocessors $\Omega = \{\rho_1, \rho_2, \cdots, \rho_m\}$
Ensure:
  Returns schedule S if one could be found, else return an error Ω

$P \leftarrow \varnothing$
for $\rho_i \in \Omega$ **do**
  $P \leftarrow P + (\tilde{f}_i^{max}, \rho_i)$
end for
$S \leftarrow \varnothing$
for $\tau_i \in \Gamma$ **do**
  $pmax(max(P_1)$
  if $p_0 < \tau_i$ **then**
    return ERROR
  end if
  $p \leftarrow (p_0 - \tau_i, p_1)$
  $P \leftarrow P + p$
  $S \leftarrow S + s$
end for
return S

A. Min-Core Worst-Fit Algorithm

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.

B. Genetic Algorithm

Genetic algorithms are stochastic search techniques based on Darwin's "Theory of Evolution" [26]. They take an initial population of solutions and use mutation and crossover to evolve a better final solution. HyWGA use a mixed population with the result from the MW algorithm combined with randomly generated schedules.

VI. Results and Discussion

In this section we describe our testbed [22] and confirm the power model used by running a series of tasks on the testbed cluster and compare the actual power consumption with the calculated theoretical power consumption. We also present data to show the effectiveness of our novel Branch-and-Bound Algorithm (Alg. 1) compared to previously suggested Hybrid Min-Core Worst-Fit Genetic Algorithm [13].

A.

We use a testbed cluster at

To confirm the power model we run both algorithms on 2 different task sets where U_{tot} is 25 and n is 250 and 450 respectively generating 4 different schedules. Using our power model we can calculate the theoretical power consumption

for the schedules. We then compare this with the actual measured power consumption by running the schedule on our testbed. We set up the testbed experiment by generating a single task for each processor based on the total utilization on that processor. A period of 10s is used such that if a processor has a 50% utilization a 100% utilization program is ran for 5s starting every 10th second. Momentary power consumption is sampled every 0.5s and the task is ran for 5 periods. Average power consumption is then $\frac{P_{tot}}{100}$, where P_{tot} is the total sampled power consumption. The result from this experiement show that our power model is accurate within 1.18%. It also shows that calculated difference between the HyWGA and BaB-A is maintained with a maximum error of 7.28% (Figs. 1(a)-1(d)).

With the power model confirmed we compare the performance of BaB-A and HyWGA by generating a set of task sets, applying the algorithms and comparing the schedules. Maximum temperature is randomly selected as a value between $40 - 50°$C. 9 different task sets are generated with the maximum utilization U_{tot} set to 25, 35, and 45 and the number of tasks n set to 250, 350, and 450 respectively. Maximum task size is used to increment U_{min} in BaB-A (Alg. 1). Each task set is run 10 times each to find a standard error, standard deviation is used to determine the standard error. Results are shown in Figs. 2. We can see a clear improvement of BaB-A over the HyWGA, with a difference up to 3.25%. For the task sets with a $U_{tot} = 45$ the improvement is between 0.36 and 0.74% (Figs. 3(a)-3(c)), to further investigate why this was we look at the testbed cluster. Due to our testbed cluster being fairly homogeneous the selection of cores becomes less important to improve the schedule, a good schedule can be found by minimizing the number of cores used. The maximum improvement can therefore never exceed the power consumption of adding another processor, regardless of the size of U_{tot}. This makes not only a measurement in percentage improvement misleading, as U_{tot} goes up, but can also for certain task sets give a close to optimal solution using HyWGA. By using a value of power consumption instead of percentage for improvement difference we get improvements ranging from 3.64 to 21.99W. This result is still not good enough for the minimum improvement. To see if the problem is with the cluster being too homogeneous we tweak the testbed, making it more heterogenenous, this should then give a more general performance improvement to our algorithm. The results shown in Figs. 2(a)-2(c) show that this is the case with performance improvements ranging from 23.92 to 45.43W, a much more general improvement. This shows that the power of our algorithm lies in it being dynamic enough to handle clusters where there are significant differences in processors.

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.

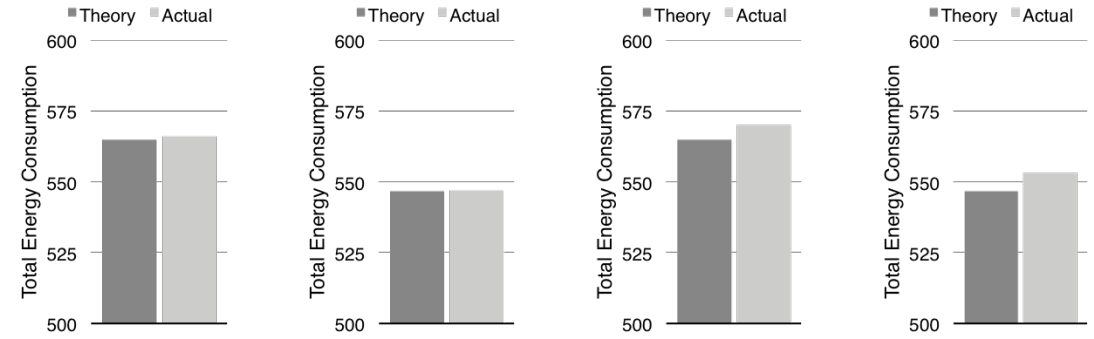
[22] S. Li, S. Wang, T. Abdelzaher, M. Kihl, and A. Robertsson, "Temperature aware power allocation: An optimization framework and case studies," *Sustainable Computing: Informatics and Systems*, vol. 2, no. 3, pp. 117–127, 2012.

[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.

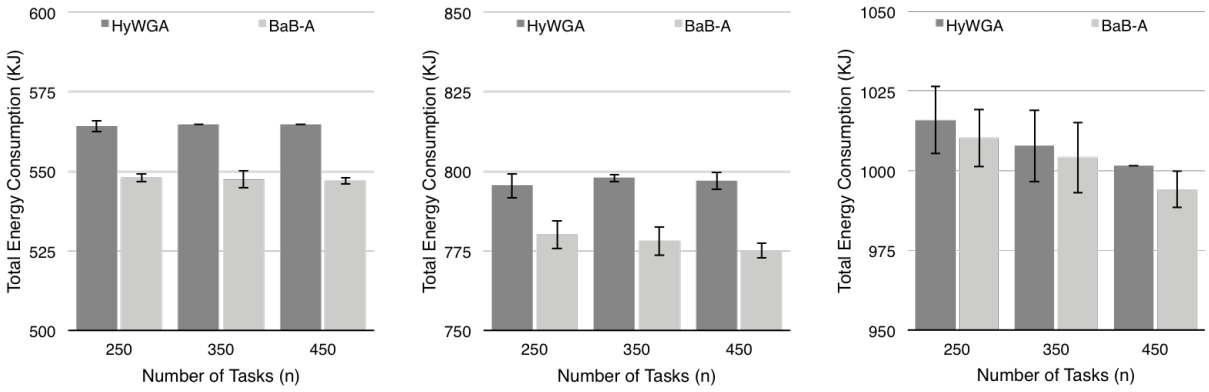
[25] J. A. Stankovic, M. Spuri, M. D. Natale, and G. C. Buttazzo, "Implications of classical scheduling results for real-time systems," *Computer*, vol. 28, pp. 16–25, 1995.

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



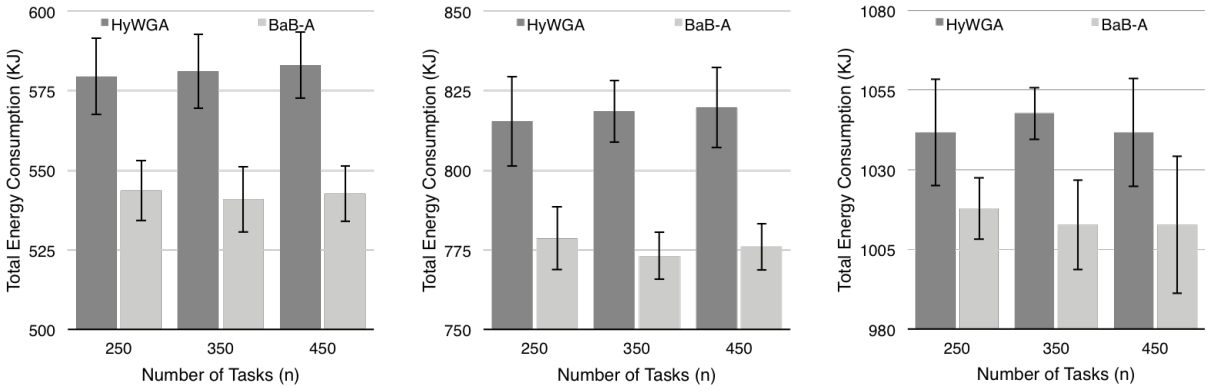
(a) $U_{tot} = 25, n = 250$ HyWGA (b) $U_{tot} = 25, n = 250$ BaB-A (c) $U_{tot} = 25, n = 450$ HyWGA (d) $U_{tot} = 25, n = 450$ BaB-A

Fig. 1. Theoretical vs. Actual Power Consumption on Testbed Cluster



(a) $U_{tot} = 25$ (b) $U_{tot} = 35$ (c) $U_{tot} = 45$

Fig. 2. Theoretical Total Power Consumption for Testbed Cluster



(a) $U_{tot} = 25$ (b) $U_{tot} = 35$ (c) $U_{tot} = 45$

Fig. 3. Theoretical Total Power Consumption for Heterogeneous Cluster