March 2015

Reliabilibity Aware Thermal Management of Real-time Multi-core Systems

Shikang Xu
University of Massachusetts Amherst

RELIABILITY AWARE THERMAL MANAGEMENT OF REAL-TIME MULTI-CORE SYSTEMS

A Thesis Presented

by

SHIKANG XU

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING

February 2015

Electrical and Computer Engineering

RELIABILITY AWARE THERMAL MANAGEMENT OF REAL-TIME MULTI-CORE SYSTEMS

A Thesis Presented

by

SHIKANG XU

Approved as to style and content by:

Israel Koren, Co-chair

C. M. Krishna, Co-chair

Sandip Kundu, Member

C. V Hollot, Department Chair
Electrical and Computer Engineering

ABSTRACT

RELIABILITY AWARE THERMAL MANAGEMENT OF REAL-TIME MULTI-CORE SYSTEMS

FEBRUARY 2015

SHIKANG XU

B.Sc., SICHUAN UNIVERSITY

M.S.E.C.E., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Israel Koren and Professor C. M. Krishna

Continued scaling of CMOS technology has led to increasing working temperature of VLSI circuits. High temperature brings a greater probability of permanent errors (failure) in VLSI circuits, which is a critical threat for real-time systems. As the multi-core architecture is gaining in popularity, this research proposes an adaptive workload assignment approach for multi-core real-time systems to balance thermal stress among cores. While previously developed scheduling algorithms use temperature as the criterion, the proposed algorithm uses reliability of each core in the system to dynamically assign tasks to cores. The simulation results show that the proposed algorithm gains as large as 10% benefit in system reliability compared with commonly used static assignment while algorithms using temperature as criterion gain 4%. The reliability difference between cores, which indicates the imbalance of thermal stress on each core, is as large as 25 times smaller when proposed algorithm is applied.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

1.1 Objective of Research

Continued scaling of CMOS technology has led to processors with large density of transistors and much more powerful computation ability. Researchers have had to deal with the resulting side effects of scaling, namely the increasing working temperature of VLSI circuits. High temperature can lead to a greater probability of permanent errors (failure) in VLSI circuits. While these failures increase cost in term of replacement of chips in general-purpose systems, they can also lead to more severe consequences in real-time systems, especially hard real-time systems. Improving the reliability of VLSI circuits has become a focus of current research. With multi-core architectures gaining in popularity, it is worthwhile to look into ways of reducing the thermal-induced failure rates in a multi-core processor.

Another kind of failure that threatens a hard real-time system is transient failure. A transient failure is an error caused by sudden and extreme changes in the environment (a temperature spike or large amount of radiation) but can recover after the environment returns to normal. Such transient errors are very hard to predict. The reliability referred in this research work focuses on the reliability with respect to the permanent failure.

The objective of this research is to develop a real-time task assignment and scheduling algorithm for multi-core processors, which ensures that deadlines are met while reducing the thermally induced failure rate thereby improving the reliability of the system.

1.2 Contributions of This Work

This research studies and presents an adaptive scheduling approach where workloads are dynamically assigned to multi-cores to deal with thermal stress in real-time systems. It is shown that dynamic assignment of workloads using estimated reliability as a criterion, rather than temperature, enhances the system reliability.

1.3 Thesis Organization

In Chapter 2, a literature study on previous research in thermal-aware computing is presented. Chapter 3 provides a brief introduction to reliability models of VLSI circuits and real-time system scheduling. Chapter 4 presents the motivation of this work, the reliability models chosen, systems models and the proposed reliability aware scheduling algorithm. In Chapter 5, the simulation structure and the benefit of proposed reliability aware scheduling algorithm over previously developed algorithms is presented. In Chapter 6, the conclusion is drawn and possible direction of future work is proposed.

CHAPTER 2

PREVIOUS WORKS

2.1 Works in Temperature-Aware Scheduling

While reducing the power/energy consumption is an indirect approach to lower the processor temperature, many researchers use temperature as the primary criterion. In [24] and [25], a temperature control method for general purpose multi-core system was developed in order to keep the maximum temperature of the processor under a given threshold while minimizing the power consumption. The method uses a table computed offline and an online speed selection phase to keep maximum temperature under a given threshold. The table is obtained by simulating the thermal behavior of a processor and contains the clock frequencies at which each core should operate under different temperature and performance requirements. The online phase is triggered periodically to choose the clock frequencies according to the current maximum processor temperature and performance requirement.

In [15], the authors propose a frequency control method which minimizes the completion time of the last finished task while operating under temperature constraints. This research assumes that each core is assigned with a specific task whose power consumption and execution cycles are known and the algorithm computes frequency for each core during execution based on this information.

In [31], the authors propose a task allocation and scheduling algorithm for platform-based MPSoC design and co-synthesis MPSoC design. The workload consist of a task graph, which has a period and deadline. In this research, tasks are allocated and scheduled according to criticality. The criticality of a task is calculated based on its

position in the task graph, WCET and power or temperature impact. The authors of [18] claim that according to their algorithm, the thermal-aware approach outperforms the power-aware approach in terms of maximal and average temperature reduction.

In [13], an offline task allocation and speed assigning algorithm to optimize the throughput under thermal constraints is proposed. In this research, the processor is assumed to have multiple ideal cores (speed can be selected anywhere from 0 to full speed) and each core can have independently selected speed. First, the authors show that the thermal flow in the horizontal direction can be neglected since the horizontal thermal resistance (i.e. resistance to heat flow in the horizontal plane) is four times higher than the vertical thermal resistance and the cores are usually surrounded by cache which is relative cooler. The maximum acceptable speeds of each core running each task are first computed by simulation. The tasks are then assigned to cores such that the sum of the speeds of the cores is maximized.

In [23], the authors report a thermal balancing policy to reduce the temperature gradient across the multi-core processor via task migration. The tasks will be migrated from hotter to cooler cores when the temperature difference between them is greater than a threshold.

In [10], the researchers also proposed a set of task migration policies to reduce the temperature gradient across the multi-core processor. In this research, there is a predefined preferred working condition (mean temperature, temperature gradient and peak temperature) for the processor. During execution, these three parameters will be compared with the preferred condition and the differences (negative or positive) will be weighted and added up. This value will be accumulated over time and when it reaches a threshold, the task migration will be performed.

Some work has also been reported on thermal aware computing for real-time systems. There are two intuitive ideas when considering the processor temperature. One is to adjust the workload of a core/processor according to its temperature. The

simplest way to do this is to let the core/processor run until the temperature reaches a threshold and stop or slow it down for a while to let it cool down to keep the temperature from rising further [14]. The second approach is to balance the load of each core/processor so that the temperatures of the cores can be balanced and a hotspot on the die can be avoided [9].

In [9], different approaches to assigning tasks to processors are compared in order to minimize the instantaneous temperature of a multi-core processor. These methods include First-Fit, Next-Fit, Best-Fit, Worst-Fit and the one proposed by the authors, Largest Task First (LTF). In this research, the periodic real-time tasks are statically assigned to each core according to the partitioned methods. The tasks on each core are then scheduled using the Earliest Deadline First (EDF) algorithm. Simulation results of this research show that LTF partitioning is probably the best from a thermal viewpoint.

In [11], a global scheduling approach is developed for sporadic real-time tasks to minimize the peak temperature in a system with a homogeneous multi-core processor. The speed of each core in [11] is derived offline, according to the system performance requirement (e.g. the sum of speeds of cores should be large enough to finish tasks before their deadline and the core with highest speed should be able to finish the task with largest utilization) using non-linear programming to minimize the peak temperature. The cores will always run at these speeds. The tasks can be scheduled according to global (tasks are allowed to execute on any available core) EDF[5] or Deadline Monotonic (task with a smaller interval between arrival time and deadline will have a higher priority) algorithm (DM)[6].

In [14], a speed and voltage assignment method to minimize the completion time of the last finished task while meeting temperature constraint and deadlines of all tasks is developed. In this research, the authors first optimize the finish time of the last finished task subjected only to a temperature constraint. They adopt the result from

[9] that the optimal speed of core is full speed when temperature is under threshold, 0 when temperature is over threshold and some value between 0 and full speed when the temperature is equal to the threshold. By finding the speed that can maintain temperature below the threshold, the authors solve the problem when no deadline constraint exists. When a constraint on deadline is added, the proposed method will deal with deadline misses from the first time when such a miss happens and search if a speed change in some previous time intervals (time interval is defined as the time between when a deadline is missed and the finishing time of tasks before this deadline) can solve this while keeping the temperature under threshold. However, this method is designed for the situation where the number of cores equals to the number of tasks, namely, each core is only assigned with one task.

2.2 Works in Reliability-Aware Scheduling

In the past few years, instead of treating reliability improvement as an indirect benefit of thermal management, researchers have started to consider the reliability of VLSI circuit directly and try to optimize it. In [22] and [20], the authors develop a dynamic model for electro-migration (the model will be discussed in Section 3). In these researches, the authors proposed that the previously suggested speed scheduling policies are conservative compared with scheduling according to their reliability model and reliability constraints. They claim that sometimes, the processor can run in a faster speed since low speed interval will compensate for this in terms of reliability. In [33], the authors discuss the failure in VLSI circuits brought about by oxide breakdown and proposed a method to adopt static(in terms of temperature) reliability models to dynamic working condition. The authors also proposed a workload prediction scheme for server based on past workload of a server and pre-defined system life time. Based on the dynamic reliability and workload prediction, the author designed a PID

controller for DVFS such that the performance (in terms of supply voltage), can be improved without violating the given reliability constraint.

CHAPTER 3

BACKGROUND IN VLSI CIRCUIT RELIABILITY AND REAL-TIME SYSTEMS

3.1 VLSI Circuit Reliability

The reliability of VLSI circuit is affected by multiple mechanisms. Modeling these has been an active research topic for decades. In this section, the mechanism and quantitative models of oxide breakdown, electro-migration (EM) and thermal cycling are briefly introduced.

3.1.1 Oxide Breakdown

Oxide breakdown is also known as dielectric breakdown. It is a mechanism that leads to a low resistance path in an oxide insulating area. The effect of oxide breakdown on modern VLSI circuits is getting worse due to thinner oxide layers brought about by technology scaling. The present understanding of the oxide breakdown process can be illustrated by Figure 3.1 [29].

When a voltage in applied across the gate oxide, there is an electron flow when the electrical field over the oxide is high enough. In thick oxides, the current is controlled by Fowler-Nordheim tunneling, while the current is generated by quantum-mechanical tunneling in thin oxide ($<$ 3nm)[29]. The electrons across the oxide trigger several defect generation processes including impact ionization and anode holes injection (which occur at high voltages), hole trapping and hole-related defect generation.

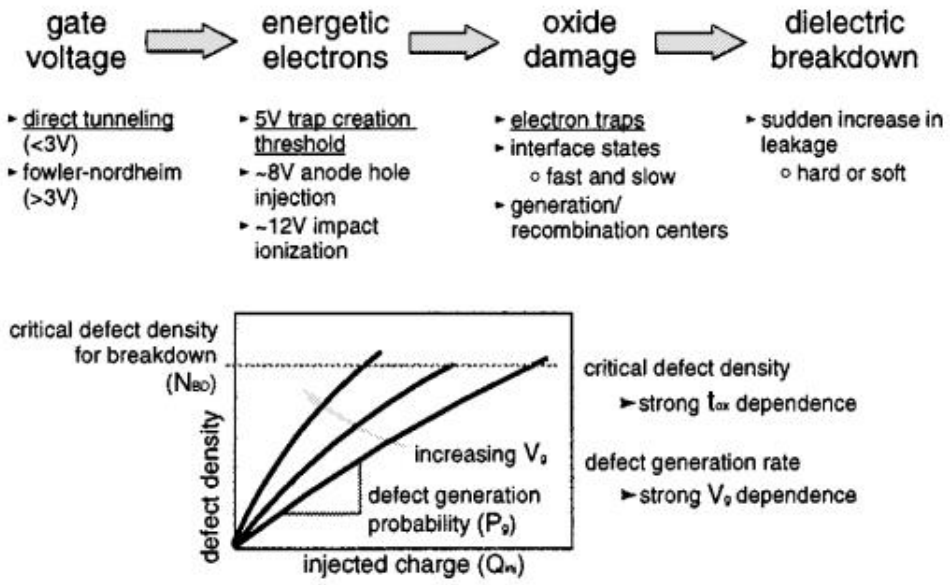


Figure 3.1: Mechanism of defect generation to breakdown ([29])

According to [28], the Mean-Time-To-Failure (MTTF) due to Oxide-Breakdown process is modeled as:

$$MTTF_{bd} = A_{bd} * (\frac{1}{V})^{(a-bT)} * e^{\frac{X+(Y/T)+ZT}{kT}} \tag{3.1}$$

where V is the voltage applied to the gate oxide, T is the absolute temperature in Kelvin, k is Boltzman constant and A_{bd} is scale factor. The values of other parameters are from RAMP: a=78, b=-0.0081,X=0.759eV, Y=-66.8eV*K and Z=-8.37e4 eV/K.

3.1.2 Electro-Migration (EM)

EM is a failure caused by the movement of metal atoms in wires due to the electric field of a VLSI circuit. The movement causes an open or short circuit in a wire. The movement of the metal atoms is mainly caused by the electrical field, temperature gradient and diffusion process in the wire [16]. The mechanism behind EM has been

studied for a long time. One of the early results that is still widely used for estimation was proposed by J. Black in 1967 [7]

$$MTTF_{em} = A_{em} * J^{-n} e^{\frac{E_a}{kT}} \tag{3.2}$$

where A_{em} is scale factor, k is Boltzman constant and T is absolute temperature in Kelvin. J is the current density. According to [21], the worst case current density is 1e6 A/cm^2. E_a is activation energy and is 0.9eV for copper. n is material based constant and is 1.1 for copper [27].

3.1.3 Thermal Cycling

Thermal cycling is a mechanical stress mechanism that is manifested in many locations in VLSI circuits including solder connections and thin-film interfaces [16]. When the temperature of the circuit changes, the components of the circuit expand or contract at different rates due to different material compositions. These differences in expansion/contraction rate eventually lead to a problem with adhesion. This situation gets worse when dynamic power management techniques such as Dynamic Voltage Scaling(DVS) are employed, since the temperature of the circuit can be expected to vary much more frequently [16]. In [27], thermal cycling is modeled as:

$$MTTF_{tc} = A_{tc} * (\frac{1}{T - T_{ambient}})^q \tag{3.3}$$

where A_{tc} is scale factor, $T_{ambient}$ is the ambient temperature in Kelvin and q is the Coffin-Manson exponent, an empirical determined material dependent constant.

3.2 Real-Time Systems

3.2.1 Tasks In Real-Time Systems

Differs from general purpose systems, a real-time system must process information and produces a response before a specified time (deadline of a task). Thus, the tasks

in real-time systems have characteristics different those in general purpose systems. Tasks in real-time systems are classified into periodic and aperiodic tasks.

Most of the tasks in real-time systems are periodic tasks and are executed periodically. The periods(time interval between two consecutive arrivals of a task) and worst case execution times (WCET) of periodic tasks in real-time systems are known at designed time. Usually, the deadlines are equal to the periods. This means that each task iteration should be finished before the next iteration is released. In practice, the execution time of one iteration of a task is often much smaller than the WCET. The actual execution time depends on the input, cache misses and other factors and is generally unknown in advance.

Aperiodic tasks are those whose periods are not know in advance. One special category of tasks in aperiodic tasks is sporadic task. The time interval of two consecutive arrival of a sporadic task is not known at design time but has a minimum value.

3.2.2 Task Scheduling In Real-Time Systems

Since real-time systems require tasks to be finished before deadline, the research on task scheduling problem is important. The first goal of task scheduling for a real-time system is to ensure that a task can be finished before its deadline. For real-time systems with a single processor or single-core processor, we have a well-developed scheduling theory [18]. Among the algorithms for single processor or single-core processor real-time system, Earliest Deadline First (EDF), in which tasks with earliest deadlines have higher priority, has been proved to be an optimal algorithm for a periodic workload with deadlines equal to periods; it can successfully schedule all such task sets if the total utilization of tasks is under 1.0 if there is no preemption overhead. Also, techniques like Dynamic Voltage Scaling (DVS) have been studied and implemented for saving energy [32].

For real-time systems with multiprocessor or multi-core processors, the scheduling problem becomes complex because task assignment needs to be taken into account. A **partitioned** (non-migration) algorithm is one where all iterations of a task can only execute on the same pre-assigned core while a **global** (full migration) algorithm is one where one iteration of a task can be preempted and then be resumed on a different core. There is also **restricted** migration algorithm, under which one iteration of a task must execute on the same core, but different iterations of a task may execute on different cores.

For real-time systems with multiprocessors or multi-core processors, EDF is no longer an optimal algorithm although it is widely used [4]. A sufficient condition for partitioned EDF to successfully schedule a task set on a m-core processor is that the sum of utilizations U_{sum} should be bounded by $\frac{\beta m+1}{\beta+1}$[19], where β equals to $\lfloor \frac{1}{U_{max}} \rfloor$ and U_{max} is the utilization of the task with the largest utilization. The sufficient condition for global EDF to successfully schedule a task set is $U_{sum} < m - (m-1)U_{max}$[12].

CHAPTER 4

SYSTEM RELIABILITY AWARE SCHEDULING ALGORITHM

4.1 Reliability Models

Electro-Migration and Oxide-Breakdown are reported to be dominant mechanisms that will lead to permanent failure of VLSI circuits as CMOS technology scales [26]. Thus, this research focuses on the reliability with respect to the electro-migration and oxide breakdown.

The failure of a system is a random process and the reliability of a system at time t is translated as the probability that the system is functional through time interval $[0, t]$.In [33], the authors stated that the probability of device failure occurrence can be represented by the Weibull distribution (other distribution can be used to replace Weibull distribution to calculate reliability if proved to fit better):

$$F(t) = 1 - R(t) = 1 - e^{-(t/\alpha)^{\beta}} \tag{4.1}$$

where $F(t)$ is the failure occurrence probability, $R(t)$ is the reliability function, β is Weibull slope parameter(=2,[30]) α is a scale parameter[33] and $\alpha = MTTF/\Gamma(1 + 1/\beta)$.

The reliability models mentioned above are obtained from a combination of experimental results and proposed physical models. They usually model the reliability of circuits under constant temperature and the MTTFs are the mean time to failure if the temperature remains constant. But in practice, the working environment of a

processor is dynamic, as is the temperature of a running processor. In this research, the approach of [33] is adopted to get the reliability in a dynamic thermal environment : divide time into k time frames, [0,Δt],[Δt, 2Δt],[(k-1)Δt, kΔt]. In each time frame, the temperature and voltage are steady, the reliability of a functional block in processor is:

$$R_{blk}(t) = R(k\Delta t) = \prod_{i=1}^{i=k}[1 - (R_i((i-1)\Delta t) - R_i(i\Delta t))] \tag{4.2}$$

where $R_i(i\Delta t)) = R_{i_{em}}(i\Delta t)) * R_{i_{bd}}(i\Delta t))$. $R_{i_{em}}(i\Delta t))$ and $R_{i_{bd}}(i\Delta t))$ are the reliability due to electro-migration and oxide breakdown and are calculated by equation 4.1 using the MTTFs get from the reliability model with the temperature of the ith time interval.

The reliability of a core at time t is the product of the reliability of all the functional blocks of the core at time t.

4.2 Motivation

As is stated in the previous section, the Weibull distribution is used to model the reliability of VLSI circuit. As shown in Figure 4.1, the reliability of the VLSI circuit will decrease faster as it ages. It is easy to prove:
In a dual-core system, for a given task set (energy consumption is fixed), keeping the reliability of cores the same all the time (keep the temperature of two cores the same all the time) will maximize the reliability of the systems when each core is regards as a single thermal node and the reliability of core is calculated using electro-migration mode.

Assume the total energy consumed by a certain set of tasks in interval [0,t] on a dual-core system is J. If the total energy is uniformly consumed and equally divided, the power on each core is:

$$P_0 = J/2t \tag{4.3}$$

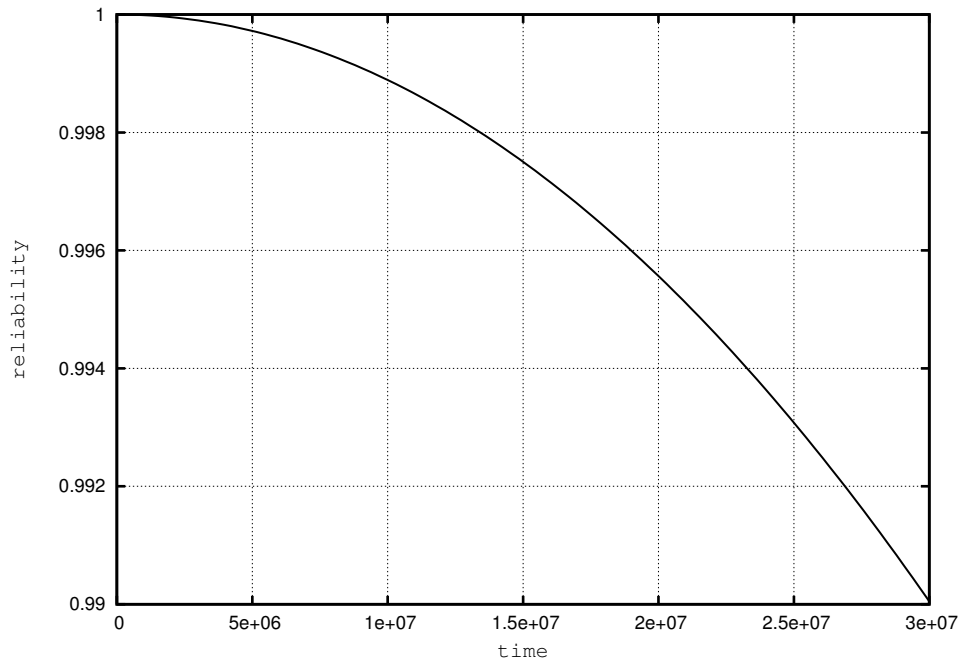


Figure 4.1: Reliability Decreases(β=2,α=3e8)

Using the standard thermal-equivalent circuit models [17], in which getting heat flow equilibrium temperature is similar to electric circuit equation. Thermal heat flow, thermal resistance, thermal capacity and temperature are analogous to electrical current, resistance, capacity and voltage respectively. The temperature at time t will be:

$$T_0 = T_{amb} + P_0 * R - (T_{init} - (T_{amb} + P_0 * R)) * e^{-\frac{t}{RC}} \tag{4.4}$$

If two cores do not have identical power, instead, P_1 is the power of core 1 and P_2 is the power of core 2. $(P_1 * t + P_2 * t = J)$

$$T_i = T_{amb} + P_i * R - (T_{init} - (T_{amb} + P_i * R)) * e^{-\frac{t}{RC}} \tag{4.5}$$

where, i=1 or 2. From above equation, we can see, $2T_0 = T_1 + T2$.

Using the equation 4.1 and 3.2, the reliability at time t is:

$$R(t) = exp(-C * t^{\beta} * e^{-\frac{E_a \beta}{kT}}) \tag{4.6}$$

where $C = (\frac{1}{A*\Gamma(1+1/\beta)*J^{-n}})^{\beta}$

When two cores have identical and uniform power through [0,t], the reliability of the dual-core system is:

$$R_{sys}(t) = exp(-2 * C * t^{\beta} * e^{-\frac{E_a\beta}{kT_0}}) \tag{4.7}$$

when the two cores have power P_1 and P_2 respectively, the reliability of the system is:

$$R_{sys}(t) = exp(-C * t^{\beta} * (e^{-\frac{E_a\beta}{kT_1}} + e^{-\frac{E_a\beta}{kT_2}})) \tag{4.8}$$

Thus, to compare the reliability of the above two situation (two cores with same power and different power), it only needs to compare $e^{-\frac{E_a\beta}{kT_1}} + e^{-\frac{E_a\beta}{kT_2}}$ and $2 * e^{-\frac{E_a\beta}{kT_0}}$ while $2T_0 = T_1 + T_2$. It is easy to observe $e^{-\frac{E_a\beta}{kT_1}} + e^{-\frac{E_a\beta}{kT_2}} - 2 * e^{-\frac{E_a\beta}{kT_0}} \geq 0$ and this proves the statement.

4.3 System Models

In this research, all cores in the multi-core system are assumed to share a main memory and higher level caches. All cores have the same speed and no Dynamic Voltage and Frequency Scaling (DVFS) technique is applied. With this assumption, the overhead of task migration due to context switches is greatly reduced.

Tasks will be assigned to each core according to its utilization before execution starts. During execution, this assignment will be adjusted according to the algorithm introduced below. The restricted migration model is followed: a given task iteration will stay on the same core for its entire execution; however, different iterations of the same task may be assigned to different cores. Tasks and different iterations of the same task are also assumed to be independent.

Based on the above assumption, task migration is assumed to have no overhead in this research and the scheduling algorithm will not consider that the overhead of task migration will lead to deadline misses.

All tasks are assumed to be periodic . Each task τ_i is characterized by the worst case execution time (WCET) e_{τ_i} and the period p_i. The relative deadline (the interval from the arrival of an iteration to the time it should be finished) of τ_i is equal to the period p_i. Thus the utilization of τ_i, U_i, is equal to $\frac{e_i}{p_i}$.

The reliability of the processor will be the product of the reliability of each core. For example, in a dual-core system, the reliability of the systems is $R_{sys} = R_{c1} * R_{c2}$.

4.4 Algorithm Description

As is proved in section 4.1, the reliability will be maximized when the cores are always thermally balanced. However, such balancing at all times is not practical since different tasks will have different thermal impact. Thus the reliability aware scheduling is proposed to dynamically adjust the reliability of each core.

As mentioned earlier, Largest Task First(LTF) was shown in [9] to be able to reduce the temperature difference among cores more efficiently than the other offline partitioning algorithm. The initial task assignment will follow the LTF algorithm.

As a workload executes, the reliability of each core will be updated periodically. Using equation 4.2, it only needs the temperature profile since last reliability calculation and last updated reliability to obtain the current reliability. The workloads on each core will not be adjusted when there is a small difference in reliability, because the small difference may not exist or even be reversed by the time the next job arrives. Instead, the reliability decreasing rate of each core m (the reliability difference of core m between two consecutive updates), δ_m, will be monitored. The proposed algorithm will update the accumulated difference in the reliability decreasing rate between each pair (m, n) of cores, γ_{mn}, which is defined by equation 4.9 :

0 **Notation:**
τ_i: a task in the real-time system,with WCET e_i,period p_i
C_j: core in system
L_j: utilization of core **j** (sum of utilizations of jobs assigned to core **j**)
TSS_i: steady state temperature of τ_i

0 **Initial_Assign()**

1 sort tasks in descending order of utilization;
2 assign[i]=-1 for i=0,1..,;
3 L_j=0 for j=1,2,...,m;
4 **for** each task i
5 j=the # of core with min{U};
6 assign[i]=j;
7 $L_j += e_i/p_i$;
8 **if**$(L_j > 1)$ return failure message;

Figure 4.2: Pseudocode for Initial Task Assignment

$$\gamma_{mn}(t) = \sum_{t=t_l}^{\bar{\tau}} (\delta_m(t) - \delta_n(t)) \tag{4.9}$$

where t_l is the time when the workload on either core m or core n is adjusted and $t_0 = 0$. $\bar{\tau} = \arg\min_\tau \{\sum_{t=t_l}^{\tau} (\delta_m(t) - \delta_n(t)) > thresh \wedge workload\, adjusted\}$. m and n are two integers denoting core m and core n.

The initial value of γs are 0. If γ_{mn} is positive, the reliability of core m decreases faster than the reliability of core n and vice versa. When the γ_{mn}s are updated, the maximum absolute value of γ_{mn} will be checked to see if it exceeds the threshold. If so, the workload adjustment algorithm will be triggered. Workload adjustment includes migrating/swapping tasks between the core pair of which the γ_{mn} is being checked (if $\gamma_{mn} > 0$, core n is considered to be more reliable, vice versa). If the load adjustment (task migration or task swapping that will be introduced below) fails due to high load (large utilization) on the cores, the pair of cores with the second largest absolute value of γ_{mn} will be chosen. Then, a check is made to see if the this quantity exceeds the threshold. If so, task migration or task swapping is performed. If not, the execution will continue with current task assignment. If the task migration or task swapping also fails on the pair of cores with the second largest absolute value of γ_{mn},

the pair with third, fourth,.... largest value of $abs(\gamma_{mn})$ will be chosen to perform load adjustment. This process will continue until load adjustment (task migration or task swapping) is successfully performed on one pair of cores or the pair of cores chosen has a $abs(\gamma_{mn})$ smaller than threshold. Every time the load adjustment is triggered, only the load on one pair of cores will be adjusted. And all the γs related two this two cores will be set to 0.

The first load adjustment that will be tried is migrating the task from the less reliable core to the more reliable core with. The migration here indicates that the next job of the migrated task will be executed on the target core. The current job of the task will stay on the core it is originally assigned to. In order to guarantee that all tasks meet their deadlines, the utilization of the target core should be smaller than or equal to 1 after migration.

When the utilization of each core is close to 1, it is quite possible that no task can be migrated from one core to another. This can also happen when the total utilization of the task sets are slightly higher than 1, but the task migration mechanism wants to move all tasks from one core to another. When this is encountered, the task assignment will stay unchanged and a task swapping process will be activated when there is no utilization space to migrate a task.

In the task swapping process, each core has a task list in which tasks are sorted according to its $U_i * Tss_i$ (Tss_i is steady state temperature of task i). The steady state temperature of task i is the steady temperature when it is executed repeatedly with $U_i = 1.0$. The task with the maximum $U_i * Tss_i$ on the less reliable core will be chosen to swap with the task with the minimum $U_i * Tss_i$ on the more reliable core. This swapping may fails if the utilization of the task on the less reliable core is larger than the utilization of the task on more reliable core and lead to the utilization of the less reliable core larger than 1.0. If so, the swapping process will try to swap two tasks with the minimum $U_i * Tss_i$ on more reliability core with the one task from the

less reliable core. If it still fails, then the swapping process will try two swap three tasks with the minimum $U_i * Tss_i$ on the more reliable core with the one task from the less reliable core. Until all tasks on the more reliability core are included. If the task with the maximum $U_i * Tss_i$ can not be swapped after the above process, the task with second maximum $U_i * Tss_i$ on the less reliable will be chosen to repeat the process, until all tasks on the less reliable core are tried. It is also possible that the task from the more reliability core has a larger utilization and the above tries fail. Then a reverse process will be activated. The task with the minimum $U_i * Tss_i$ from the more reliable core is first chosen to swap with the tasks with maximum $U_i * Tss_i$. If it fails, then the algorithm will try to swap the first two tasks with maximum $U_i * Tss_i$ on the less reliable core with the tasks with smallest $U_i * Tss_i$ on the more reliable ... and so on.

```
0  online_load_adjustment()
1  Update all γ_mn;
2  Define S as the set containing all core pairs
3  (j,k)=the core pair in S with maximum abs(γ);
4  while(S is not empty)
5    if(abs(γ_{j,k})>= Threshold)
     // assume core_j is less reliable
6      if(migrate(j,k))
7        all γs related two core j and core k are set to 0;
8        break;
9      else
10       if(swap(j,k))
11       all γs related two core j and core k are set to 0;
12       break;
13       else
14         remove (j,k) from S;
15         (j,k)=the core pair in S with maximum abs(γ);
16   else
17     continue execution with current assignment;
18     break;

0  migrate(m,n)
1  τ_i: lowest-utilization task on core m;
2  if((L_n + e_i/p_i) <= 1)
3    assign[i]=k;
4    L_n+ = e_i/p_i;
5    return 1;
6  else return 0;

0  swap(m,n)
1  for each task i in on core m // start from the task with the maximum U_i * Tss_i
2    k=1
3    while k <= No. of tasks on core n
4      try swap task i with the first k task(s)
       with the minimum U * T_ss
5        successes: break,return 1
6        fails: k++
         //fails if utilization of one core is larger than 1 after swapping
7  for each task i in on core n // start from the task with the maximum U_i * Tss_i
8    k=1
9    while k <= No. of tasks on core m
10     try swap task i with the first k task(s)
       with the minimum U * T_ss
11       successes: break,return 1
12       fails: k++
13 return 0;
```

Figure 4.3: Pseudo Code for Online Load Adjustment

CHAPTER 5

SIMULATION AND RESULT

5.1 Simulation Structure

A simulator of a processor with two Alpha 21264 cores is built to evaluate the proposed reliability aware algorithm. Temperature of each functional block is needed to calculate the reliability of the processor. The temperature is calculated using TILTS, which is a faster thermal simulation method than the popular HotSpot tool [2] (the accuracy and speed up of TILTS compared with HotSpot can be found in [3]). The power profile of the workload is obtained using Wattch [8] simulator.

In simulation, the exact thermal information (temperature) of each functional block can be obtained and is used to get the reliability of the processor. This quantity is used to access the proposed algorithm using the thermal information from practical approaches (i.e. task migration decision will be made according to thermal information from practical approach; reliability used to compare with other algorithm is obtained using exact information). In practice, the temperature of each functional blocks is obtained using the power profile estimated from the performance counter or temperature sensors. But there may not be enough performance counters/sensors for all functional blocks. To model this, a simplified floorplan of Alpha 21264 (Figure 5.2) is used to get the temperature needed for reliability calculation. When calculating temperature using the simplified floor plan, the power of each block in the simplified floorplan is equal to the sum of the power of the blocks of the original Alpha 21264 floorplan that is contained in the block of simplified floorplan. The task migration in

the proposed algorithm will be based on the reliability estimated using the simplified floorplan.

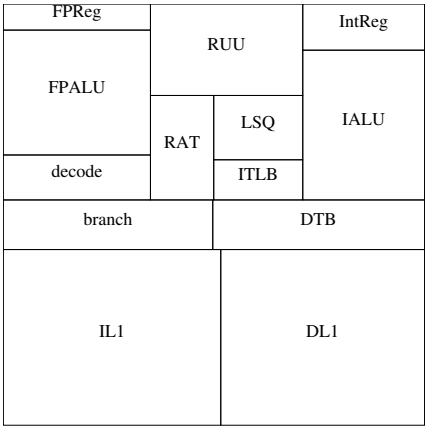


Figure 5.1: Original Alpha 21264 Floorplan[1]

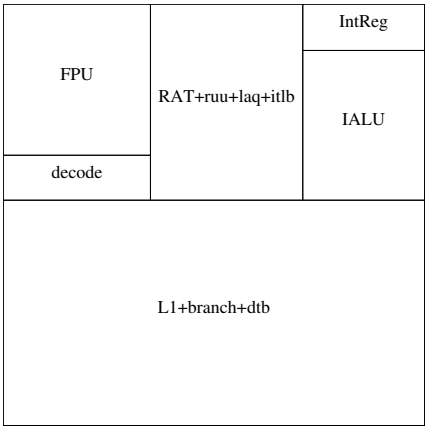


Figure 5.2: Simplified Floorplan

The workloads are real-time tasks made of benchmarks from SPEC2K(Table 5.1). Benchmarks act as periodic real-time tasks which are independent of each other and are characterized by worst case execution time (WCET) and period (equal to relative deadline). The WCET of each task is the time running the corresponding benchmark in Wattch simulator. The power profile of each benchmark is sent to thermal simulator when the corresponding task is in execution. In real-time system, the actual execution time will not always be equal to WCET. The actual execution time of each task is modeled as a $rand * WCET$ where $rand$ is a random variable belongs to (0:1.0] and follows normal distribution. This random variable is generated during the simulation of execution. (e.g. If the actual execution time of a task is 0.5 of its WCET, the power of every two time steps in power profile will be averaged and read as the power of one time step. Thus, the power changes in the execution of this task are captured and the execution time is scaled according to the ratio to WCET). In practice, a task can have different thermal impact on the system when its utilization is different.

This is modeled by running multiple task sets in which the tasks are the same but the utilization of each task are randomly generated.

Table 5.1: Benchmarks Used in Simulation

Benchmark	Benchmark Description
Bzip	Integer; Compression
Gcc	Integer; C Complier
Gap	Integer; Group Theory
Mesa	Floating Point; 3-D Graphics
Mgrid	Floating Point; Multi-grid Solver
Sixtrack	Floating Point;High Energy Nuclear Physics Accelerator Design
Galgel	Floating Point; Computational Fluid Dynamics

5.2 Simulation Results

To evaluate the proposed algorithm, it is compared with three previously developed scheduling algorithms:

Utilization balancing scheduling is a static scheduling which assigns tasks to each core using largest task first assignment (LTF) before execution [9] using the worst case execution utilization (WCET/Period). No migration will be performed during execution.

Instantaneous temperature based scheduling is the most straight forward thermal aware scheduling. The task migration will be triggered when the instantaneous temperature difference between two cores is larger than a threshold (10 degree in the comparison).

Temperature history based scheduling will record the temperature difference (positive or negative) between cores/processors. When the accumulated temperature difference reaches a threshold, tasks will be migrated from the historically hotter core to the historically cooler core [10]. Initially, tasks are assigned according to LTF assignment. During execution, the difference (positive or negative) between maximum temperatures of each core will be accumulated after each temperature calculation. If

the accumulated value is larger than a threshold (5000 in current simulation), the tasks on the hotter core will be moved to the cooler core. The value 5000 is set randomly. If the threshold is large, the scheduling will act more like utilization balancing algorithm. If it is smaller, it will act more like the instantaneous temperature based algorithm.

The utilization balancing scheduling will be regarded as the baseline of comparison. The "R - U" stands for the benefit of the proposed reliability aware algorithm over utilization balancing scheduling. The "T_inst - U" stands for the the benefit of instantaneous temperature based scheduling over utilization balancing scheduling. The "T - U" stands for the the benefit of temperature history based scheduling over utilization balancing scheduling. The benefit is calculated using function:

$$Benefit_{R-U} = (R_R(t^*) - R_U(t^*))/(1 - R_U(t^*)) * 100\% \tag{5.1}$$

$$Benefit_{T\_inst-U} = (R_{T\_inst}(t^*) - R_U(t^*))/(1 - R_U(t^*)) * 100\% \tag{5.2}$$

$$Benefit_{T-U} = (R_T(t^*) - R_U(t^*))/(1 - R_U(t^*)) * 100\% \tag{5.3}$$

where t^* is the time when reliability of the system using utilization balancing algorithm is equal to 0.99. The parameters for the two reliability models is shown in Table 5.2.

Table 5.2: Parameter Value of Reliability Models

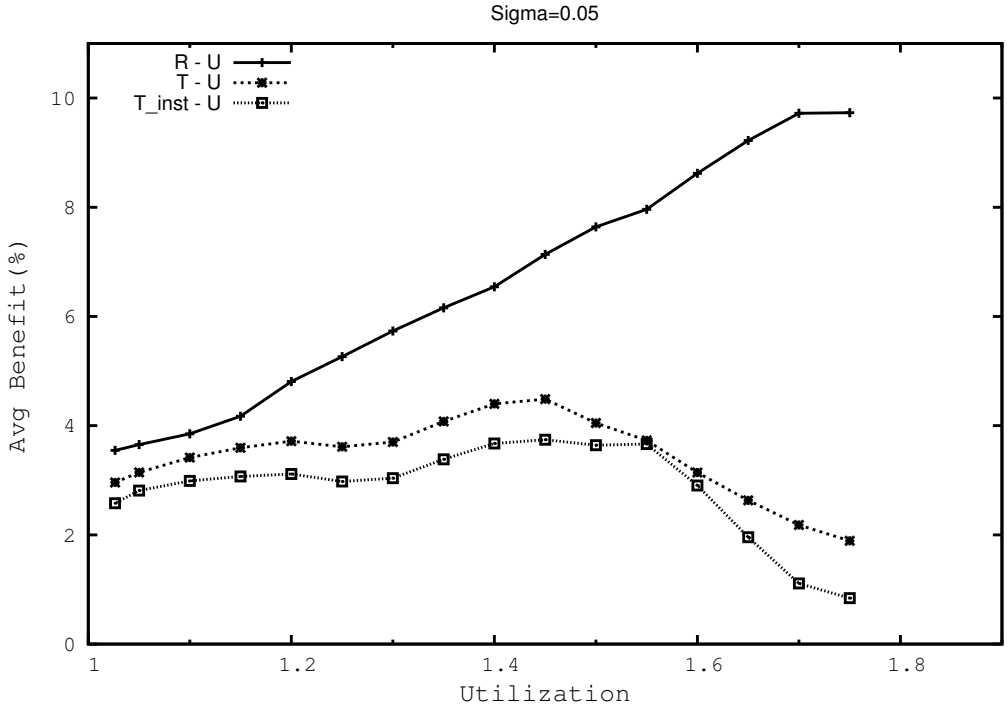
EM		Oxide Breakdown	
A_{em}	0.3	A_{bd}	2.88e7
J	1e6A/cm^2	a	78
E_a	0.9eV	b	-0.0081
n	1.1	X	0.759eV
		Y	-66.8eV*K
		Z	-8.37e4 eV/K

Figure 5.3 show the average benefit of different scheduling algorithms over the baseline algorithm running multiple randomly generated task sets. The x-axis is the total utilization (worst case) of the system while the y-axis is the benefit using the benefit functions presented above.

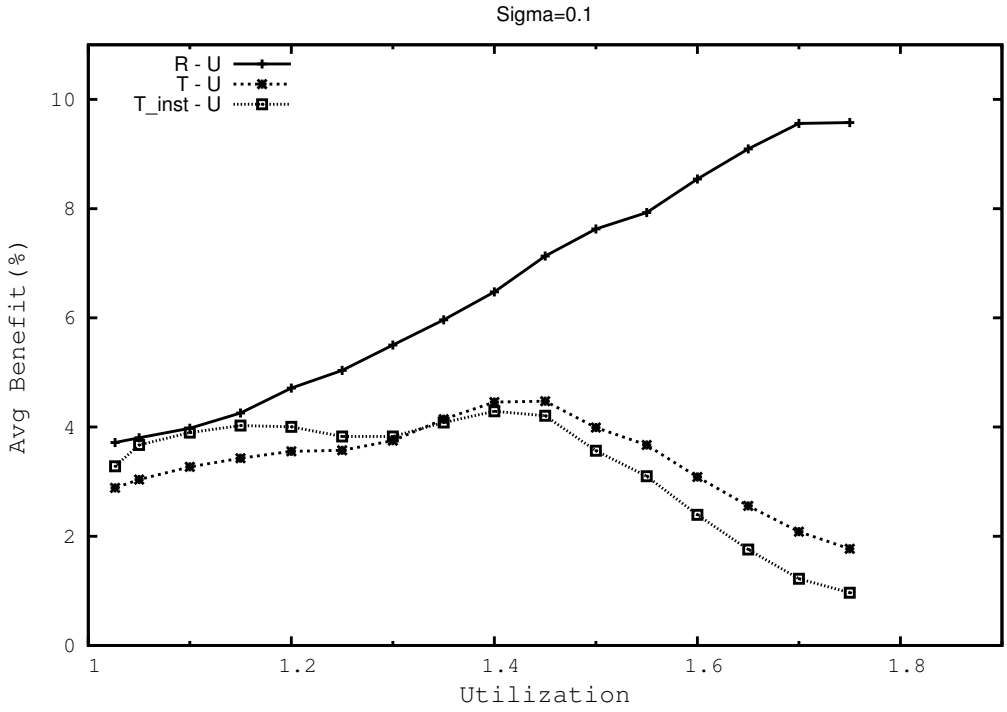
As is shown in Figure 5.3, when the total utilization of the system is high([1.6,1.8]), the benefit is larger. This is due to the fact that the the thermal load imbalance is higher when the system has less idle time. Using the task swapping mechanism, the proposed reliability aware algorithm can still adjust the thermal load on cores. The proposed reliability aware algorithm doesn't gain too much over the two comparisons when the utilization is smaller. This is because the static utilization balancing algorithm has already properly balanced the thermal load on two cores. In Figure 5.3, it can be seen that, as the actual execution time of each task varies larger (larger sigma) from its mean execution time, the benefit also goes down. This is because the reliability aware algorithm can be seen as predict the future thermal impact based on the historical thermal impact. As the actual execution varies, this prediction is less precise.

In Section 4.2, it is proved that the reliability of the whole system will be maximized when the reliability of each core is balanced. Figure 5.4 shows the average reliability difference using different algorithms. The task sets used are the same task sets used for results in Figure 5.3. As is shown, the proposed reliability aware algorithm gives much smaller reliability difference than other three. If no dynamic frequency/voltage scaling is performed, this benefit can be seen as the maximum possible benefit.

As is mentioned in Section 5.1, a simplified floorplan is used to get the temperature and reliability used for task migration/swapping since there may not be enough temperature information in a real processor. Figure 5.5 shows the difference in benefit if the original floorplan of Alpha 21264 core is used to get the temperature and

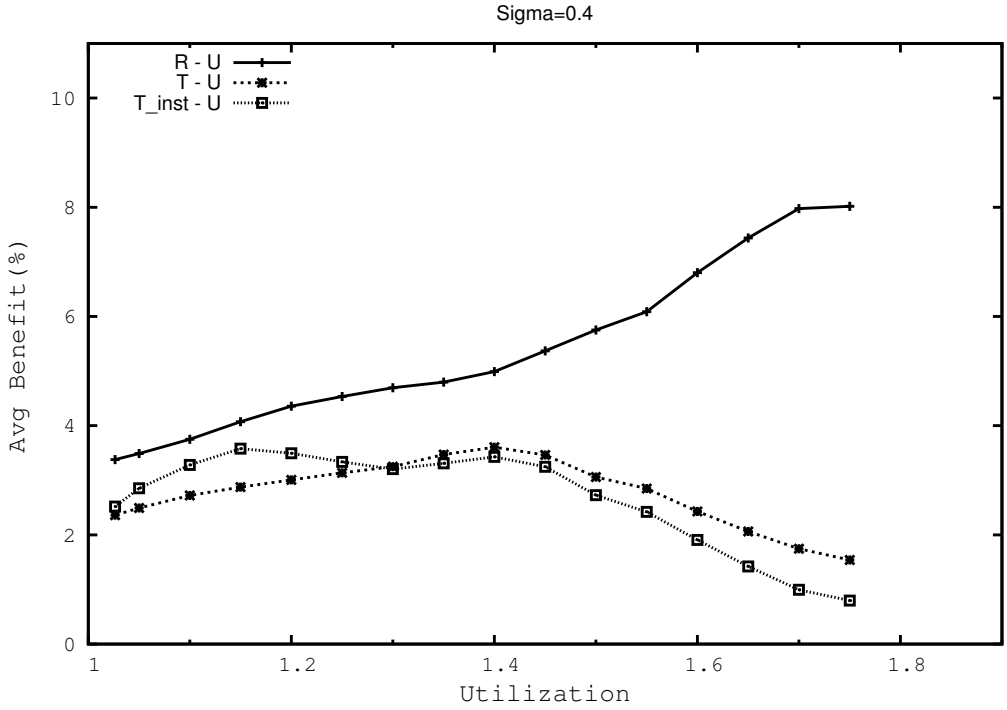


(a)

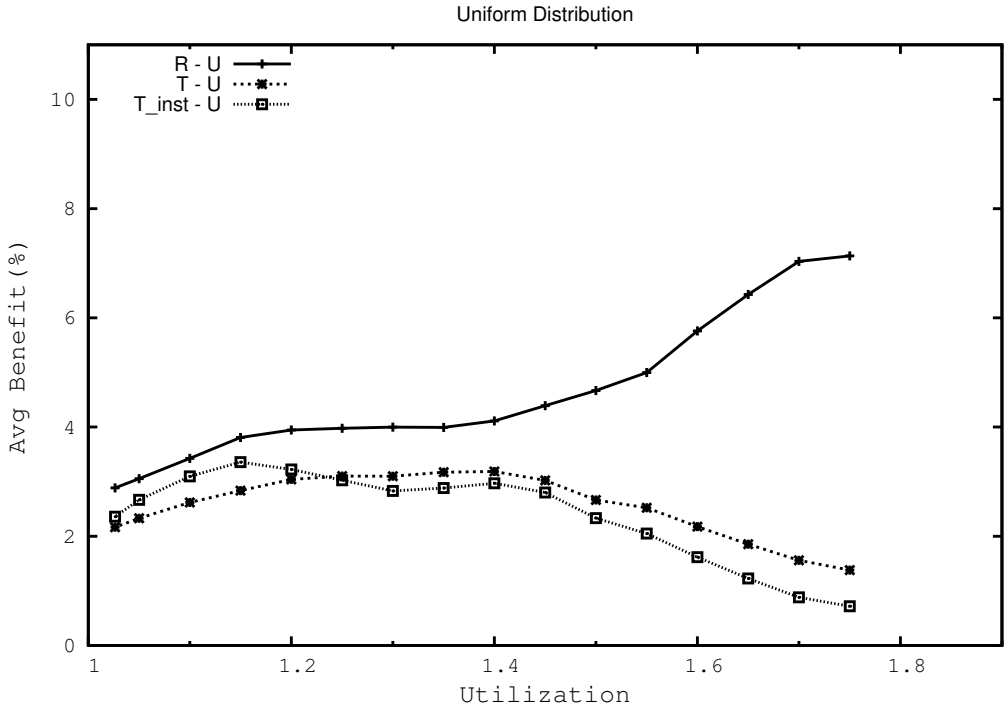


(b)

Figure 5.3: Average Benefit over Static Algorithm

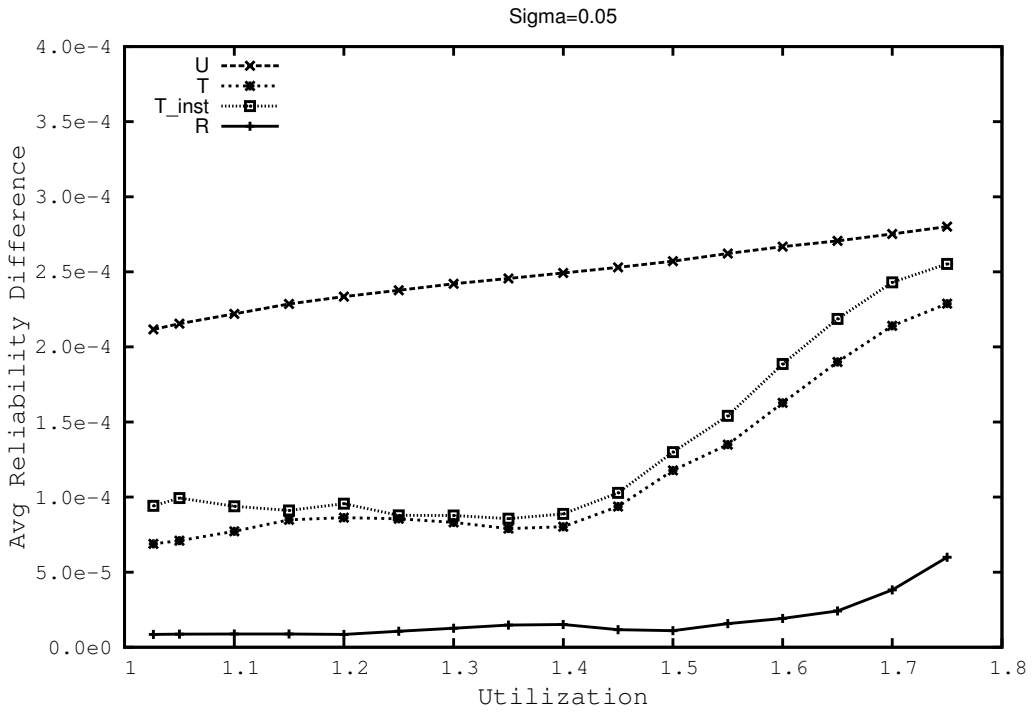


(c)

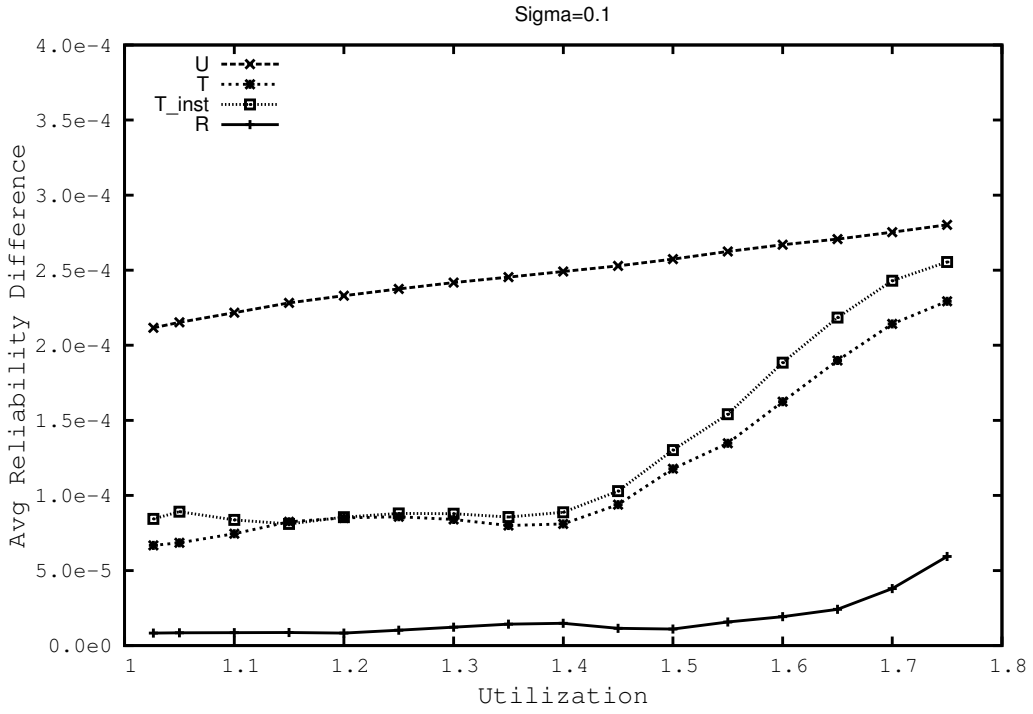


(d)

Figure 5.3: Average Benefit over Static Algorithm(Continued)

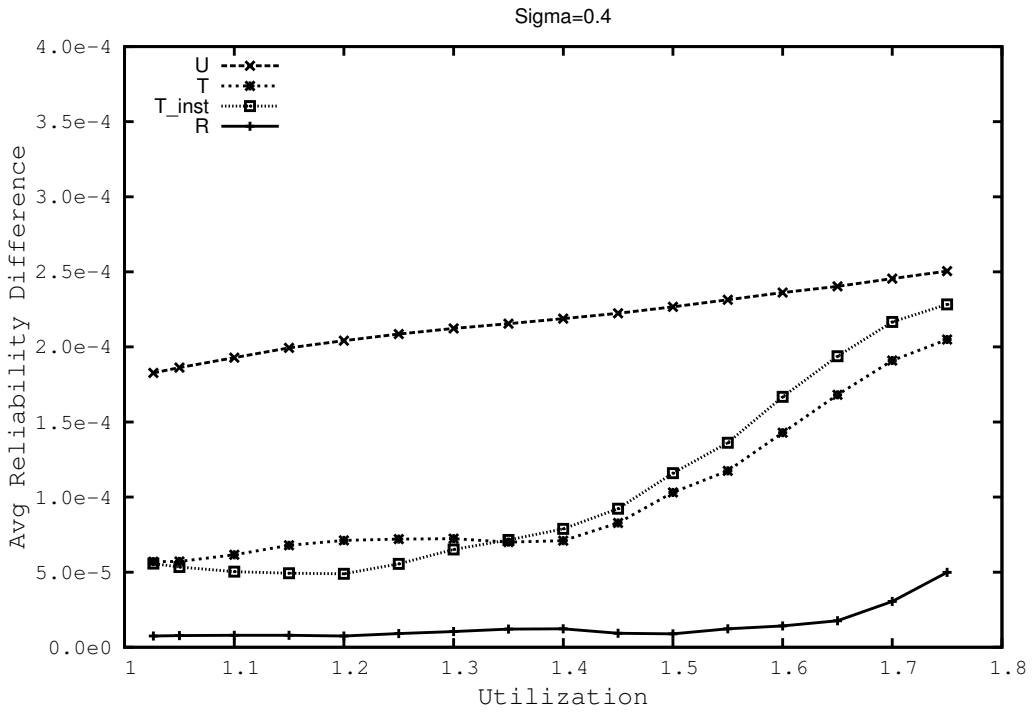


(a)

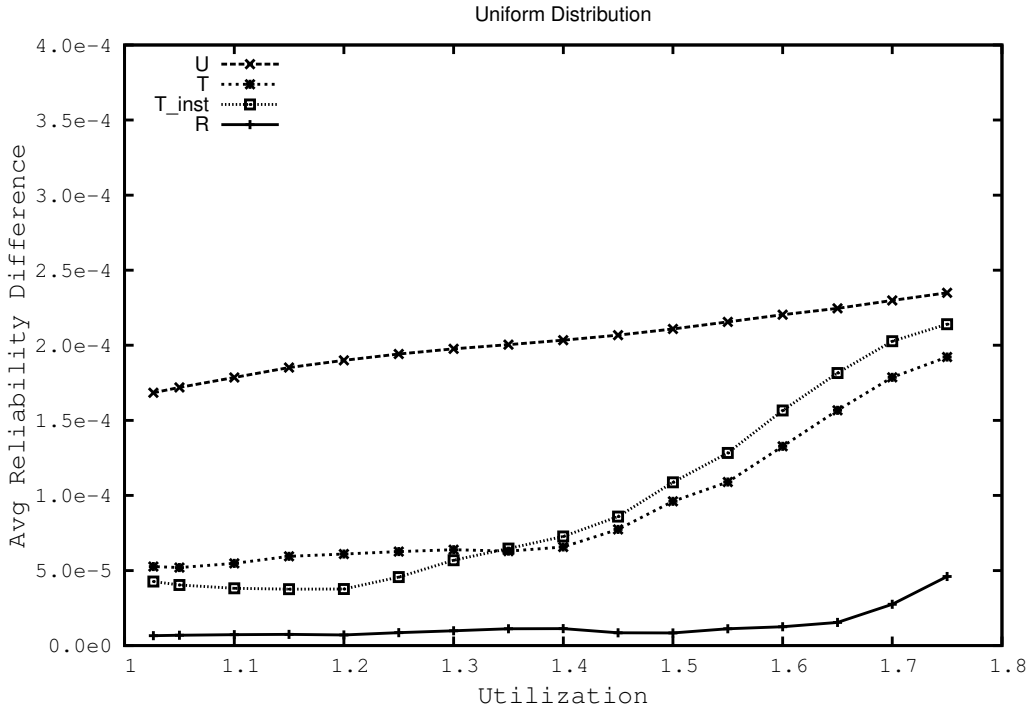


(b)

Figure 5.4: Average Reliability Difference Among Cores when Using Different Algorithms(Continued)

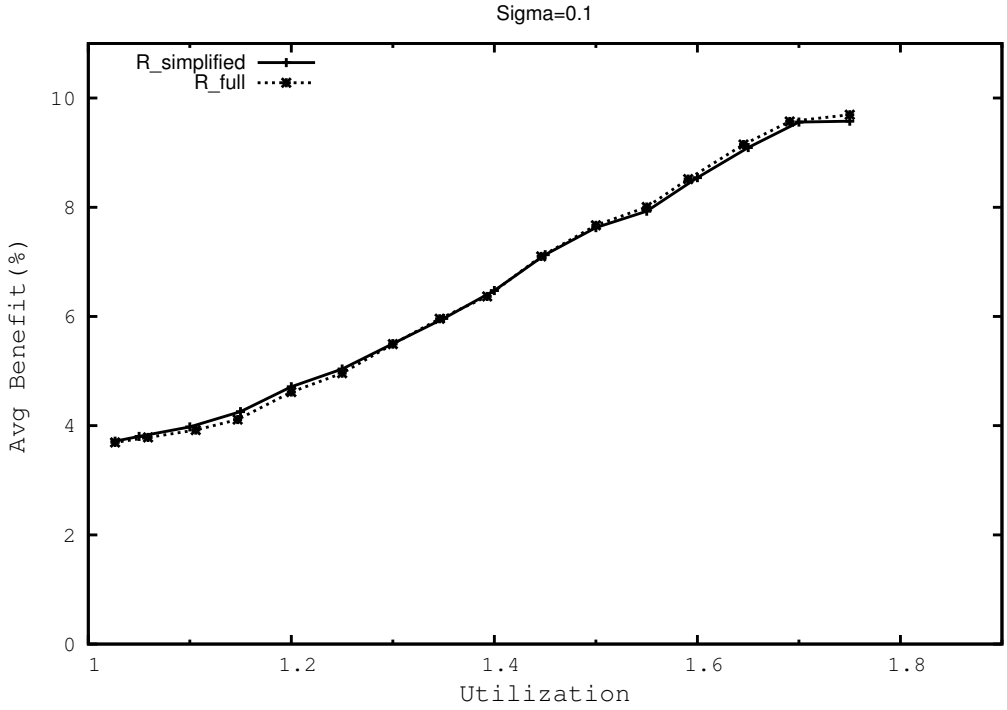


(c)

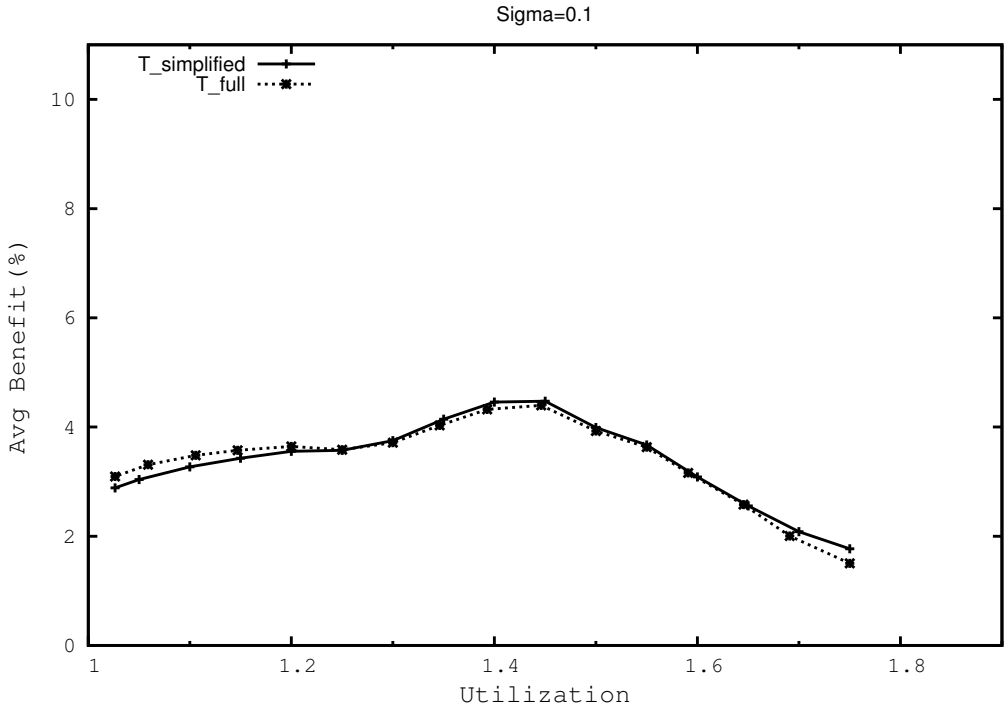


(d)

Figure 5.4: Average Reliability Difference Among Cores when Using Different Algorithms(Continued)



(a)



(b)

Figure 5.5: Benefit over Static Algorithm when Using Different Floorplans

reliability. In Figure 5.5 , "R_simplified" and "T_simplified" stand for the benefit of reliability aware algorithm and temperature history based scheduling algorithm using the simplified floorplan (this is the data shown in Figure 5.3). "R_full" and "T_full" stand for the benefit using the original floorplan.

As is shown in Figure 5.5, there is only a very small difference using different floorplans. This is because the reliability of a core is dominated by the hot blocks. The integer ALU and integer register are individual blocks in the simplified floorplan. These two blocks are usually the hottest parts of the core. The other two hot blocks, the floating point register and floating point ALU, are merged in a simple block. Since the activities of these two blocks are tightly related, the temperature of this two block will vary in the same direction. Merging this two blocks into one will not lead to the situation where hot block become a cool block after being merged.

The reliability update interval is another parameter that can affect the performance of the proposed algorithm. Reliability of each core is updated using the average temperature of past interval. A small interval will lead to an increase in the overhead due to temperature acquirement and reliability calculation. Using a large interval, on the other hand, the algorithm will have less opportunities to migrate tasks and may fail to capture the impact of hot but small tasks. The benefit shown above uses an interval of 0.5 second. Figure 5.6 shows the difference in benefit when using different reliability update interval. The benefit decreases when the reliability update interval is larger. This is because the temperature difference between two cores used to calculated reliability is small when the interval is large. The estimated reliability is not precise enough. The reliability difference is not enough to trigger the task migration/swapping. Less migration/swapping will be performed.

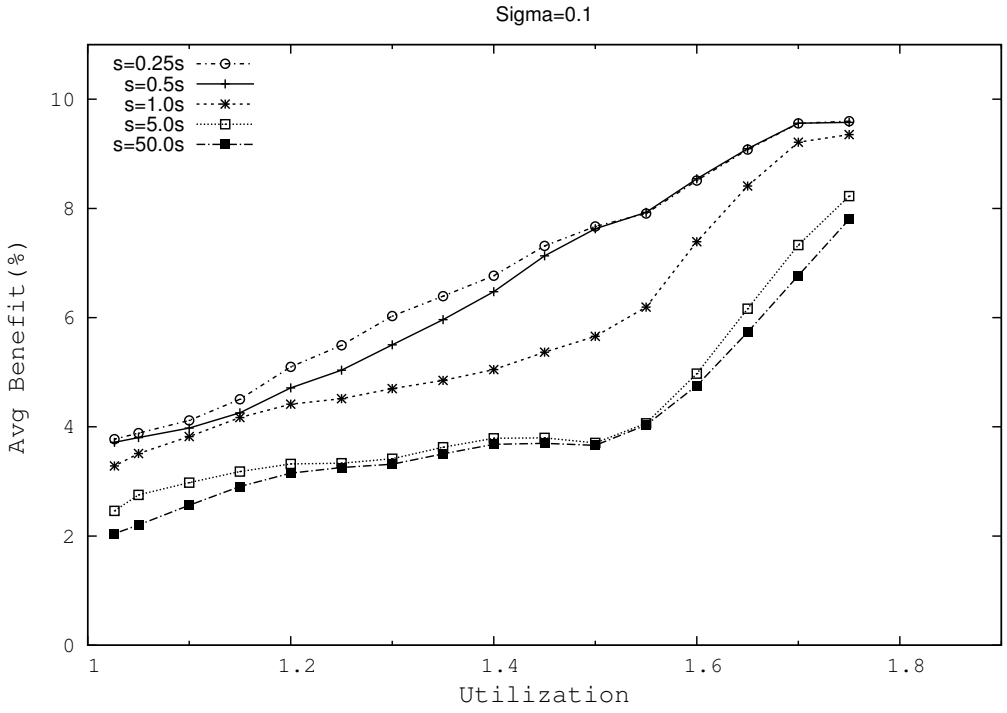


Figure 5.6: Benefit Using Different Reliability Update Interval

CHAPTER 6

CONCLUSIONS

This research proposes a reliability aware dynamic workload balancing algorithm to improve the reliability of multi-core real-time systems. The simulation result shows that the proposed algorithm outperforms the static scheduling algorithm and the previously suggested algorithms that balance the load on each core according to temperature. The proposed algorithm gains nearly 10% of benefit in reliability over the static algorithm while algorithms using temperature information gain 4% or less over the static algorithm. Using the proposed reliability aware algorithm, the reliability difference between two cores is 5 to 25 times smaller (high utilization to low utilization) than static approach and 4 to 10 times smaller than the temperature based algorithm. It is proved in this research that keeping the thermal impact balanced on the cores at all times (the reliability difference among cores to be 0 all the time) will maximize the system reliability. The reduction of the reliability difference among cores achieved by the proposed algorithm shows the proposed algorithm provides a considerable improvement over the previous algorithms.

The suggested direction of future study is to reduce the reliability difference among cores further. For a given task set with certain energy consumption, it is possible to use DVFS technique to make the power of each core fluctuate less (thus the temperature of core will fluctuate less) and achieve a further reduction in the reliability difference among cores.

APPENDIX

RANDOM GENERATION OF TASK UTILIZATION AND ACTUAL EXECUTION TIME

In this research, the proposed reliability aware algorithm need to be evaluated under different total utilization. Also, tasks can have different thermal impact on systems. Some tasks lead to higher temperature when executing repeatedly with no break. Some tasks lead to lower temperature. Thus, the proposed algorithm is also evaluated using the task sets in which different task sets have different utilization. (For example, in task sets 1, task 1 may have a utilization of 20 percent of the total utilization but only have 10 percent of total utilization in sets 2).

The tasks used in the simulation are 7 applications from SPEC2000. The worst case execution time (WCET) of each task is the time to run it on Simplescalar with ALPHA configuration using the input from SPEC2000 package. Since WCET of each task is determined, the utilization($\frac{WCET}{period}$) of each tasks can be adjusted by setting different periods.

The utilization of each task is determined by the following way: 7 integers ($rand_i, i = 1, 2, .., 7$) in [1,100] are generated using the Mersenne twister random number generator. The utilization weight of task τ_i will be $\frac{rand_i}{\sum rand_i}$. If the desired total utilization is 1.8, the utilization of τ_i will be $\frac{rand_i}{\sum rand_i} * 1.8$. Table A.1 shows $\frac{rand_i}{\sum rand_i}$ of each task in the different task sets used for result in Chapter 5. Each column is a task sets.

In real-time system, the actual execution time will not always (seldom, actually) equal to the WCET. In simulation, this is model by setting the actual execution time of each iteration equal to the random generated ratio of WCET. This random ratio

belong to [0.1,0.2,...,1.0]. And is generated when the iteration arrives. The ratio of WCET is generated by sampling the normal distribution with different deviations.

Table A.1: Relative Contribution to Task Set Utilization

	Task Sets									
	1	2	3	4	5	6	7	8	9	10
six	0.064	0.207	0.093	0.145	0.107	0.160	0.040	0.132	0.085	0.188
gcc	0.192	0.237	0.139	0.097	0.171	0.106	0.202	0.227	0.030	0.104
mesa	0.085	0.110	0.185	0.145	0.190	0.177	0.179	0.176	0.070	0.208
mgrid	0.256	0.092	0.046	0.145	0.143	0.266	0.269	0.265	0.282	0.156
gap	0.192	0.104	0.030	0.218	0.071	0.160	0.040	0.040	0.211	0.156
galgel	0.171	0.221	0.370	0.194	0.143	0.089	0.134	0.132	0.080	0.139
bzip	0.038	0.028	0.139	0.055	0.171	0.044	0.134	0.026	0.241	0.052

BIBLIOGRAPHY

[1] Ptscalar, Dec. 2003.

[2] Hotspot, Dec. 2011.

[3] Comparison between tilts and hotspot, Dec. 2013.

[4] Baruah, S., and Carpenter, J. Multiprocessor fixed-priority scheduling with restricted interprocessor migrations. In *Real-Time Systems, 2003. Proceedings. 15th Euromicro Conference on* (July 2003), pp. 195–202.

[5] Baruah, Sanjoy, and Baker, Theodore. Schedulability analysis of global edf. *Real-Time Systems 38*, 3 (2008), 223–235.

[6] Baruah, Sanjoy, and Fisher, Nathan. Global deadline-monotonic scheduling of arbitrary-deadline sporadic task systems. In *Principles of Distributed Systems*, Eduardo Tovar, Philippas Tsigas, and Hacne Fouchal, Eds., vol. 4878 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2007, pp. 204–216.

[7] Black, James R. Mass transport of aluminum by momentum exchange with conducting electrons. In *Reliability Physics Symposium, 2005. Proceedings. 43rd Annual. 2005 IEEE International* (April 2005), pp. 1–6.

[8] Brooks, D., Tiwari, V., and Martonosi, M. Wattch: a framework for architectural-level power analysis and optimizations. In *Computer Architecture, 2000. Proceedings of the 27th International Symposium on* (2000), pp. 83–94.

[9] Chen, Jian-Jia, Hung, Chia-Mei, and Kuo, Tei-Wei. On the minimization of the instantaneous temperature for periodic real-time tasks. In *Real Time and Embedded Technology and Applications Symposium, 2007. RTAS '07. 13th IEEE* (2007), pp. 236–248.

[10] Cuesta, D., Ayala, J.L., Hidalgo, J.I., Atienza, D., Acquaviva, A., and Macii, E. Adaptive task migration policies for thermal control in mpsocs. In *VLSI (ISVLSI), 2010 IEEE Computer Society Annual Symposium on* (2010), pp. 110–115.

[11] Fisher, N., Chen, Jian-Jia, Wang, Shengquan, and Thiele, L. Thermal-aware global real-time scheduling on multicore systems. In *Real-Time and Embedded Technology and Applications Symposium, 2009. RTAS 2009. 15th IEEE* (2009), pp. 131–140.

[12] Goossens, Joël, Funk, Shelby, and Baruah, Sanjoy. Priority-driven scheduling of periodic task systems on multiprocessors. *Real-Time Syst. 25*, 2-3 (Sept. 2003), 187–205.

[13] Hanumaiah, V., Rao, R., Vrudhula, S., and Chatha, K.S. Throughput optimal task allocation under thermal constraints for multi-core processors. In *Design Automation Conference, 2009. DAC '09. 46th ACM/IEEE* (2009), pp. 776–781.

[14] Hanumaiah, V., and Vrudhula, S. Temperature-aware dvfs for hard real-time applications on multicore processors. *Computers, IEEE Transactions on 61*, 10 (2012), 1484–1494.

[15] Hanumaiah, V., Vrudhula, S., and Chatha, K.S. Performance optimal speed control of multi-core processors under thermal constraints. In *Design, Automation Test in Europe Conference Exhibition, 2009. DATE '09.* (2009), pp. 1548–1551.

[16] Karl, E., Blaauw, D., Sylvester, D, and Mudge, T. Multi-mechanism reliability modeling and management in dynamic systems. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on 16*, 4 (2008), 476–487.

[17] Krishna, C. M. Ameliorating thermally acclerated aging with state-based application of fault-tolerance in cyber-physical computers. In IEEE Transactions on Reliability (March 2015, to appear).

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

[19] Lopez, J.M., Garcia, M., Diaz, J.L., and Garcia, D.F. Worst-case utilization bound for edf scheduling on real-time multiprocessor systems. In *Real-Time Systems, 2000. Euromicro RTS 2000. 12th Euromicro Conference on* (2000), pp. 25–33.

[20] Lu, Zhijian, Huang, Wei, Lach, J., Stan, M., and Skadron, K. Interconnect lifetime prediction under dynamic stress for reliability-aware design. In *Computer Aided Design, 2004. ICCAD-2004. IEEE/ACM International Conference on* (2004), pp. 327–334.

[21] Lu, Zhijian, Huang, Wei, Stan, Mircea, Skadron, Kevin, and Lach, John. Interconnect lifetime prediction with temporal and spatial temperature gradients for reliability-aware design and run 134 time management: Modeling and applications. very large scale integration (vlsi) systems. *IEEE Transactions on* (2006).

[22] Lu, Zhijian, Huang, Wei, Stan, M.R., Skadron, K., and Lach, J. Interconnect lifetime prediction for reliability-aware systems. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on 15*, 2 (2007), 159–172.

[23] Mulas, F., Pittau, M., Buttu, M., Carta, Salvatore, Acquaviva, A., Benini, L., Atienza, D., and De Micheli, G. Thermal balancing policy for streaming computing on multiprocessor architectures. In *Design, Automation and Test in Europe, 2008. DATE '08* (2008), pp. 734–739.

[24] Murali, S., Mutapcic, A., Atienza, D., Gupta, R., Boyd, S., Benini, L., and De Micheli, G. Temperature control of high-performance multi-core platforms using convex optimization. In *Design, Automation and Test in Europe, 2008. DATE '08* (2008), pp. 110–115.

[25] Murali, S., Mutapcic, A., Atienza, D., Gupta, R., Boyd, S., and De Micheli, G. Temperature-aware processor frequency assignment for mpsocs using convex optimization. In *Hardware/Software Codesign and System Synthesis (CODES+ISSS), 2007 5th IEEE/ACM/IFIP International Conference on* (2007), pp. 111–116.

[26] Srinivasan, J., Adve, S.V., Bose, P., and Rivers, J.A. The impact of technology scaling on lifetime reliability. In *Dependable Systems and Networks, 2004 International Conference on* (June 2004), pp. 177–186.

[27] Srinivasan, J., Adve, S.V., Bose, P., and Rivers, J.A. Lifetime reliability: toward an architectural solution. *Micro, IEEE 25*, 3 (2005), 70–80.

[28] Srinivasan, Jayanth, Adve, Sarita V., Bose, Pradip, and Rivers, Jude A. The case for lifetime reliability-aware microprocessors. In *Proceedings of the 31st Annual International Symposium on Computer Architecture* (Washington, DC, USA, 2004), ISCA '04, IEEE Computer Society, pp. 276–.

[29] Stathis, J.H. Physical and predictive models of ultrathin oxide reliability in cmos devices and circuits. *Device and Materials Reliability, IEEE Transactions on 1*, 1 (Mar 2001), 43–59.

[30] Wu, E, Su, J, Lai, W, Nowak, E, McKenna, J, Vayshenker, A, and Harmon, D. Interplay of voltage and temperature acceleration of oxide breakdown for ultra-thin gate oxides. *Solid-State Electronics 46*, 11 (2002), 1787 – 1798.

[31] Xie, Yuan, and Hung, Wei-Lun. Temperature-aware task allocation and scheduling for embedded multiprocessor systems-on-chip (mpsoc) design. *J. VLSI Signal Process. Syst. 45*, 3 (Dec. 2006), 177–189.

[32] Zhu, Xiaomin, He, Chuan, Bi, Yuping, and Qiu, Dishan. Towards adaptive power-aware scheduling for real-time tasks on dvs-enabled heterogeneous clusters. In *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on Int'l Conference on Cyber, Physical and Social Computing (CPSCom)* (Dec 2010), pp. 117–124.

[33] Zhuo, Cheng, Sylvester, D, and Blaauw, D. Process variation and temperature-aware reliability management. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2010* (2010), pp. 580–585.