Workload Assignment Considering NBTI Degradation in Multicore Systems

JIN SUN and ROMAN LYSECKY, The University of Arizona
KARTHIK SHANKAR, University of Texas at Austin
AVINASH KODI, Ohio University
AHMED LOURI and JANET ROVEDA, The University of Arizona

With continuously shrinking technology, reliability issues such as Negative Bias Temperature Instability (NBTI) has resulted in considerable degradation of device performance, and eventually the short mean-time-to-failure (MTTF) of the whole multicore system. This article proposes a new workload balancing scheme based on device-level fractional NBTI model to balance the workload among active cores while relaxing stressed ones. Starting with NBTI-induced threshold voltage degradation, we define a concept of Capacity Rate (CR) as an indication of one core's ability to accept workload. Capacity rate captures core's performance variability in terms of delay and power metrics under the impact of NBTI aging. The proposed workload balancing framework employs the capacity rates as workload constraints, applies a Dynamic Zoning (DZ) algorithm to group cores into zones to process task flows, and then uses Dynamic Task Scheduling (DTS) to allocate tasks in each zone with balanced workload and minimum communication cost. Experimental results on a 64-core system show that by allowing a small part of the cores to relax over a short time period, the proposed methodology improves multicore system yield (percentage of core failures) by 20%, while extending MTTF by 30% with insignificant degradation in performance (less than 3%).

Categories and Subject Descriptors: D.4.1 [**Operating Systems**]: Process Management

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Multicore systems, negative bias temperature instability capacity rate, workload balancing, dynamic zoning, dynamic task scheduling

ACM Reference Format:
Sun, J., Lysecky, R., Shankar, K., Kodi, A., Louri, A., and Roveda, J. 2014. Workload assignment considering NBTI degradation in multicore systems. ACM J. Emerg. Technol. Comput. Syst. 10, 1, Article 4 (January 2014), 22 pages.
DOI: http://dx.doi.org/10.1145/2539124

1. INTRODUCTION

As device feature sizes continue to shrink, long-term reliability or permanent fault such as Negative Bias Temperature Instability (NBTI) affects system life-span, and leads to

A preliminary version of this article appeared in *Proceedings of the Asia and South Pacific Design Automation Conference* [2010].
J. Sun is currrently affiliated with Orora Design Technologies, Inc.
This work was partially supported by the National Science Foundation under Grant 0915537.
Authors' addresses: J. Sun, Orora Design Technologies, Inc., Issaquah, WA 98027; email: jinsun@orora.com; R. Lysecky, A. Louri, and J. Roveda, Department of Electrical and Computer Engineering, The University of Arizona, Tucson, AZ 85721; email: {rlysecky, louri, wml}@ece.arizona.edu; K. Shankar, Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX 78712; email: karthikvs@mail.utexas.edu; A. Kodi, Department of Electrical Engineering and Computer Science, Ohio University, Athens, OH 45701; email: kodi@ohio.edu.

DOI: http://dx.doi.org/10.1145/2539124

the low yield and short mean-time-to-failure (MTTF) in a multicore system [Reddy et al. 2002; Chen et al. 2003]. A number of new techniques have recently emerged to cope with NBTI aging effect using post-manufacturing burn-in strategy [Constantinides et al. 2007; Wang et al. 2007a; Bhardwaj et al. 2006]. However, very little attention has been paid to device stress and its impact on system life-span and performance in the multicore era. Device stress may happen after days of full workload operation, and requires days to relax before recovery. Letting the device completely wearout will impact the system as defective cores have to be permanently removed from the pool of active cores. A meaningful approach would be relaxing cores when they are stressed long before they are completely wearout. This approach would require solving three challenges: (1) how to assess that a core is stressed, (2) how to assign workload to relax stressed cores, and (3) how to avoid additional performance cost associated with balancing workload. The proposed approach in this article answered all these questions.

Different from existing approaches [Wang et al. 2007a; Bhardwaj et al. 2006] that focused on long-term stress using static NBTI models, we propose to use a fractional stress and recovery model to model partially stressed cores: cores alternate between a heavy workload phase (once the core has relaxed) and a light workload phase (when the core becomes stressed) with high frequency of alternation between phases. During the light workload phase, PMOS transistors of idle gates are set to "1" to relieve the stress as discussed in Abella et al. [2007]. The end result of fractional NBTI model is a core capacity rate (CR) that indicates how much additional workload one core can accept before getting over-stressed. We assume a full recovery of device in our implementation. However, the proposed algorithm is also applicable to other NBTI models, for example, the models in Paul et al. [2005] and Alam and Mahapatra [2008]. The workload changes due to NBTI-induced performance variations will follow the same modeling process.

A number of research projects have been conducted on workload scheduling and balancing with performance and reliability objectives in multicore systems. Though not solving the same problem, some of the ideas are worth mentioning and provide a good initiative of the current work. For example, Rong and Pedram [2006] formulated the problem of determining the optimal voltage schedule and task ordering as an integer linear program. In Ruggiero et al. [2006], the scheduling problem was decomposed into sub-problems of allocation and scheduling, respectively. Some other algorithms took into account temperature changes when performing task scheduling. Coskun et al. [2007], proposed a dynamic scheduling technique that incorporated the thermal history to adjust workload assignments for an optimal thermal distribution. Hung et al. [2005] presented a thermal-aware task allocation and scheduling algorithm to achieve a thermally even distribution by manipulating the peak temperature. In addition to these traditional scheduling algorithms, several recent works have focused on mitigating NBTI aging effect in multicore era. For example, Basoglu et al. [2010] proposed a predictive model to quantitatively evaluate the long-term impact of NBTI-aware task mapping and to improve device lifetime. Lin et al. [2011] developed a transmission gate-based optimization technique for minimizing NBTI-induced degradation and leakage power simultaneously.

The proposed approach has three components: (1) NBTI-introduced core performance difference estimation, (2) Dynamic Zoning (DZ), and (3) Dynamic Task Scheduling (DTS). Core performance difference is estimated using fractional NBTI model and is indicated by capacity rate. Each core has its own capacity rate. Dynamic Zoning (DZ) algorithm groups cores into zones according to core capacity rates. It starts with a rectangular region as the initial zone, and adjusts gradually considering zone connectivity and core capacity rate to match workload from the assigned flow. Then, the Dynamic Task Scheduling (DTS) algorithm allocates tasks in one zone to achieve maximum utilization with low communication cost. Figure 1 demonstrates the flow of our proposed

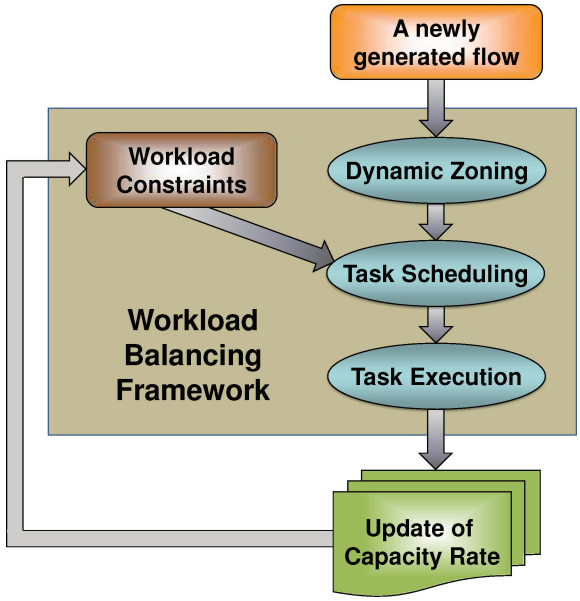

Fig. 1. The general flow of the proposed workload balancing methodology.

approach. Note that this method is iterative: the core capacity rate should be updated frequently because of changing core performance or aging effect.

Specifically, the new contributions of this article are: (1) a system NBTI stress estimation model, (2) a new workload balancing strategy considering core performance difference, (3) a novel scheduling scheme incorporating data packet size and communication cost, and (4) a new insight into the relationship between core recovery time, stress time, workload, and their impact on core life-span. This strategy has shown good system yield improvement and MTTF extension with insignificant impact on latency or communication traffic overhead. Experimental results on a 64-core system show that by allowing a part of cores (approximately 12.5%) to relax over a short time period, the proposed methodology reduces core failure rate by 20%, and extends MTTF by 30% with insignificant degradation in performance (less than 3%). Moreover, as the number of stressed cores increases, a minor increase in system degradation can be observed. It is important to mention that a number of researchers in recent years have proposed systematic approaches of designing control circuits and arranging input vectors to mitigate NBTI aging. Wang et al. [2007a], Abella et al. [2007], and Bild et al. [2009] have demonstrated effective schemes to address severe NBTI issues with minor implementation cost. The focus of this work is to balance workload adaptively under the impact of NBTI aging for the purpose of improving system reliability.

The remainder of this article is organized as follows: Section 2 introduces the fractional NBTI device and core model. Section 3 discusses the proposed workload balancing methodology considering NBTI-introduced performance difference. Experimental results are included in Section 4. Finally, Section 5 concludes the article.

2. NBTI MODEL FOR MULTICORE PERFORMANCE

NBTI limits lifetime in nano-scale integrated circuits and continues to worsen with device scaling beyond 90nm [Constantinides et al. 2007; Wang et al. 2007a; Bhardwaj et al. 2006]. When PMOS is negatively biased, the electrical field across the gate oxide produces a complicated electrochemical reaction that consequently increases the PMOS

threshold voltage over time. The impact of NBTI may take days or months to ultimately affect timing and circuit delay, eventually leading to system failure. Recent research works have confirmed that NBTI is getting worse with further scaling starting from 90-nm technology [Wang et al. 2007a].

The NBTI impact causes the core to alternative between stress and recovery phases. In general, recovery and stress periods are fairly symmetric. The NBTI-introduced threshold voltage change ΔV_{th} in stress phase follows a similar style as reported in Bhardwaj et al. [2006]:

$$\Delta V_{\text{th}} = \left(K_v \left(T(t)\right)(t - t_0)^{\frac{1}{2}} + \Delta V_{\text{th0}}{}^{\frac{1}{2n}}\right)^{2n}. \tag{1}$$

Equation (1) describes V_{th} degradation due to NBTI impact at time t, compared with the threshold voltage V_{th0} at starting time point t_0: $\Delta V_{\text{th}} = V_{\text{th}}(t) - V_{\text{th}}(t_0)$. $T(t)$ is simply the temperature fluctuation with regard to time. K_v and C are two parameters defined in compact NBTI predictive model [Bhardwaj et al. 2006]. K_v and C are both functions in terms of varying temperature $T(t)$, therefore they change over time as well. In recovery phase the corresponding V_{th} degradation can be represented as

$$\Delta V_{\text{th}} = V_{\text{th0}}\left(1 - \frac{2\epsilon_1 + \sqrt{\epsilon_2 C(T(t))(t - t_0)}}{2t_{\text{ox}} + \sqrt{C(T(t))t}}\right), \tag{2}$$

where process parameter t_{ox} denotes oxide thickness. Considering the temperature change with regard to time, we include the time dependency in parameters K_v and C, as they both are functions of $T(t)$. The complete expressions of the associated parameters in (1) and (2) can be detailed in Bhardwaj et al. [2006]. Process variability is another important factor that may exert great influence on V_{th} shift. The proposed model can be further extended to express V_{th} as a function in terms of not only temperature, but also process parameters to develop process variability-affected NBTI model.

The change in threshold voltage in turn affects device's gate delay (D_g) and leakage power from subthreshold leakage (P_{leak}). We consider gate delay first. Following Sarangi et al. [2008a, 2008b], the time required to switch a logic gate can be estimated by:

$$D_g \propto \frac{V_{\text{dd}} L_{\text{eff}}}{\gamma \left(V_{\text{dd}} - V_{\text{th}}\right)^{\alpha}}, \tag{3}$$

where V_{dd} and L_{eff} are supply voltage and transistor effective channel length, respectively, while α is a process parameter, which is typically 1.3, and parameter γ is the mobility of carriers. The path between an input and an output with the maximum delay is identified as the critical path. Note that for a single core there may be a number of critical paths. Considering NBTI-induced V_{th} shift, gate delay function can be further represented as:

$$D_g \propto \frac{V_{\text{dd}} L_{\text{eff}}}{\gamma \left(V_{\text{dd}} - (V_{\text{th0}} + \Delta V_{\text{th}})\right)^{\alpha}}. \tag{4}$$

According to (4), as V_{th} increases, D_g increases and the gate becomes slower. Therefore, V_{th} variation may deteriorate critical path delay, and may even cause severe timing errors in today's low-voltage devices.

On the other hand, although V_{th} variation has little impact on dynamic power consumption, it affects leakage power dramatically. Using an empirical leakage power model [Sun et al. 2008], the leakage power for a single transistor is given by:

$$P_{\text{leak}} = C_0 \cdot \exp\left(-C_1 L_{\text{eff}} - C_2 V_{\text{th}} + C_3 V_{\text{dd}}\right), \tag{5}$$

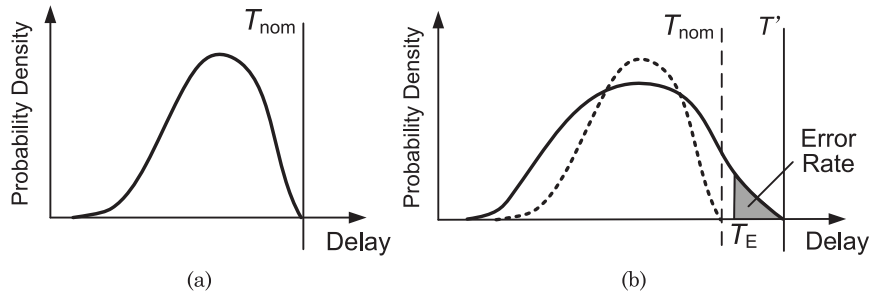

Fig. 2. Impact of V_{th} shift on core frequency (adapted from Sarangi et al. [2008b]).

where C_0 is a constant coefficient in empirical leakage model, C_1, C_2, and C_3 are the fitting exponents of transistor channel length, threshold voltage and supply voltage, respectively. Substituting the first-order V_{th} shift model $V_{th} = V_{th0} + \Delta V_{th}$ into (5), leakage power under NBTI degradation can be written as:

$$P_{leak} = C_0 \cdot \exp(-C_1 L_{eff} - C_2 V_{th0} + C_3 V_{dd}) \cdot \exp(-C_2 \Delta V_{th}) \tag{6}$$

While these equations are for a single PMOS transistor, for every gate and every core, the total leakage power is simply the summation of the leakage power of all transistors in one gate or one core:

$$P_{leakT} = \sum_j P_{leak,j}. \tag{7}$$

Here $P_{leak,j}$ denotes a single transistor's leakage power predicted by (6).

As NBTI-induced V_{th} shift causes fluctuations in leakage power and gate delay, core devices will behave differently under NBTI aging effect. Therefore, it is important to incorporate this diversity of core performance in workload assignment. Intuitively, cores with lower leakage power and shorter gate delay have the potential to accept more workload. We define a new concept of "Capacity Rate" (CR) as an indication of how much workload one core can accept, by evaluating core performance difference under NBTI aging and certain performance constraints.

We discuss the timing issue first. The V_{th} shift induces variation in gate delays, and therefore slows down some critical paths in a core device. As a result, the maximum frequency that can be achieved by the core to avoid timing errors will be lowered, since a core cannot cycle any faster than its slowest critical path can [Sarangi et al. 2008b]. If this core is not operated at the resulting lowered frequency, timing errors will occur. To estimate the rate of timing errors as a function of operating frequency, we employ the error rate model described in Sarangi et al. [2008b]. This model considers the dynamic distribution of all path delay metrics. Figure 2(a) provides an example of such distribution with no impact of V_{th} variation. All path delays are less than the nominal clock period T_{nom}, therefore, no timing error occurs. However, if NBTI aging effect is considered, V_{th} degradation may slow down some critical paths (refer to (4)). The distribution of path delays may be shown as in Figure 2(b). As delay values exceed T_{nom} on some paths, to avoid timing errors the core must be clocked with a longer period $T' > T_{nom}$. Figure 2(b) indicates that if a core operates with a shorter period $T_E < T'$, it will suffer timing errors. In other words, a core must operate at a particular frequency $f_E < f'$, where $f' = 1/T'$, to guarantee no timing error.

Such frequency loses may be reduced if core device is equipped with error detection/correction mechanisms that are capable of tolerating some variation-induced timing errors [Sarangi et al. 2008b]. Based upon this assumption, we rely on (4) to

estimate the probability of timing errors. We assume that NBTI-induced V_{th} degradation is subject to a Gaussian distribution with mean value μ and standard deviation σ. As a function of random variable ΔV_{th}, the mean and standard deviation of D_g can be obtained by using a Taylor series expansion [Papoulis 2002]:

$$\mu_{D_g} = \frac{V_{\text{dd}} L_{\text{eff}}}{\gamma} \cdot \left[(V_x - \mu)^{-\alpha} + (\alpha^2 + \alpha)(V_x - \mu)^{-\alpha-2} \cdot \frac{\sigma^2}{2} \right], \tag{8}$$

$$\sigma_{D_g}^2 = \left(\frac{\alpha V_{\text{dd}} L_{\text{eff}}}{\gamma} \right)^2 \cdot (V_x - \mu)^{-2\alpha-2} \cdot \sigma^2. \tag{9}$$

where $V_x = V_{\text{dd}} - V_{\text{th}}$. The detailed derivations are provided in Appendix. Once obtaining mean μ_{D_g} and standard deviation σ_{D_g}, we can approximately estimate the rate of timing errors. Based on the assumption in Sarangi et al. [2008b], delay metric empirically has a linear relationship with V_{th} within the parameter range of interest, and therefore approximately follows a normal distribution since V_{th} is normally distributed. This model estimates timing violation probability as the area of the shaded region in Figure 2. Alternatively, we can conveniently compute error rate using the Cumulative Probability Function (CDF) of the normalized path delays. In general, if the core is clocked with period $T_{\text{E}} = 1/f_{\text{E}} < T'$, the timing error rate is computed as:

$$ER(T_{\text{E}}) = 1 - \text{CDF}(T_{\text{E}}), \tag{10}$$

where CDF gives the cumulative probability that path delays exceed the specified clock period T_{E}. With the mean value and variance estimated by (8) and (9), such cumulative probability can be calculated based upon assumption of normal delay distribution. We set a target error rate for each core with V_{th} degradation: the induced error rate cannot exceed a threshold value ER_0. As a result, the core has to operate at a relatively lower, and safer frequency f_{E} to satisfy the timing error constraints:

$$ER(f_{\text{E}}) = 1 - \text{CDF}\left(\frac{1}{f_{\text{E}}}, \mu_{D_g}, \sigma_{D_g} \right) \leq ER_0. \tag{11}$$

Note that the timing error rate depends on not only V_{th} variation and operating frequency, but also supply voltage V_{dd}. Lower supply voltage may increase gate delays, slow down the critical paths, and eventually lead to higher error rate.

It is worth mentioning that, as device feature sizes continue to shrink, the assumption of normal distribution may not be enough for estimating timing errors in future technology. In future works, it will be necessary to extend the error rate model to consider other distribution, for example, exponential, Weibull, and even more complicated distributions. In these models, an analytical calculation of error rate based on an explicit CDF becomes unrealistic. To address this problem, the proposed NBTI model can be extended by incorporating different probability lookup tables for possible distributions. The purpose is to establish a relatively complete yet efficient table lookup scheme for error rate estimation, in order to achieve a good tradeoff between computational efficiency and complexity.

We now discuss the power issue considering NBTI degradation. The power dissipation for one core consists of dynamic power consumption and leakage power dissipation. Leakage power could account for 50% or even higher of total power of the circuit [Zhang et al. 2004]. The variation of leakage power induced by NBTI effect is predicted by (6) and (7). As known to all, the dynamic power consumption is determined by the supply voltage and operating frequency:

$$P_{\text{dyn}} = C_{\text{L}} V_{\text{dd}}^2 f, \tag{12}$$

where C_{L} is the external load capacitance, while $V_{\mathrm{dd},i}$ and f_i denote the core's supply voltage and operating frequency, respectively. Therefore, the total power consumption for one core can be described as:

$$P_{\mathrm{Total}} = C_{\mathrm{L}} V_{\mathrm{dd}}^2 f + P_{\mathrm{leakT}}, \tag{13}$$

where the total leakage power is estimated according to (7). Similarly, we set a constraint of power requirement for each core:

$$P_{\mathrm{Total},i} \leq P_0. \tag{14}$$

The power consumed by each core cannot exceed a predetermined specification. With this power limit, the operating condition $(V_{\mathrm{dd},i},\ f_i)$ of a particular core will be restricted, due to the fluctuation in leakage power caused by V_{th} degradation.

Under performance constraints, the fluctuations in leakage and delay will force the cores to operate at different points to avoid violating the limits. The cores will in turn behave in differently manners during task execution. To estimate and quantify this difference, we evaluate the task execution time on a particular stressed core, and compare it with the result on a core without V_{th} degradation. Given a specific task, we first evaluate the execution time on a nonstressed core working at full frequency, and denote it as T_{s0}. We then perform the following optimization procedure to determine the optimal execution time on the ith stressed core:

$$\begin{aligned} &\text{minimize} \quad T_{s,i}(V_{\mathrm{dd},i},\ f_i) \\ &\text{subject to} \quad ER_i(f_i) = 1 - \mathrm{CDF}\left(\frac{1}{f_i}, \mu_{D_g,i}, \sigma_{D_g,i}\right) \leq ER_0 \\ &\text{subject to} \quad P_{\mathrm{Total,i}} = C_{\mathrm{L}} V_{\mathrm{dd},i}^2 f_i + P_{\mathrm{leakT},i}(\Delta V_{\mathrm{th}}) \leq P_0, \end{aligned} \tag{15}$$

where $T_{s,i}(V_{\mathrm{dd},i},\ f_i)$ denotes the required execution time for core i running at operating condition $(V_{\mathrm{dd},i},\ f_i)$. Note that supply voltage and frequency are correlated when estimating the error rate. By solving problem (15), we can determine the optimal task execution time without violating performance constraints. We denote the optimal execution time as $T^*_{s,i}(V_{\mathrm{dd},i},\ f_i)$, which is achieved by letting the core work at its best operating condition $(V^*_{\mathrm{dd},i},\ f_i^*)$ under performance constraints. The obtained $T^*_{s,i}(V_{\mathrm{dd},i},\ f_i)$ is then compared with the execution time required for a nonstressed core with full rate. The capacity rate of the ith stressed core is determined by calculating the ratio:

$$CR_i = \frac{T_{s0}}{T^*_{s,i}(V_{\mathrm{dd},i},\ f_i)}. \tag{16}$$

The capacity rate for a core will lie in the interval [0, 1], indicating core's capability difference in workload assignment. For example, a nonstressed core requires a execution time of 30 clock cycles to complete the assigned task, while it takes 50 clock cycles for a stressed core to process the identical task. Following (16), the stressed core will have a capacity rate of 0.6, which means that it can accept at most 60% workload compared with a core with full capacity rate.

To conclude, we have transferred the NBTI degradation model to core capacity rate interpretation. The NBTI model first captures its impact on threshold voltage degradation. Considering threshold voltage degradation, we further model core's performance variation under delay and power constraints, based on which we are able to quantify core's ability in workload assignment. The end result is the capacity rate for each core in percentage. The generated capacity rate will be considered as an upper bound limit of acceptable workload on each stressed core. The workload assigned to a stressed core cannot exceed this limit, as its capacity rate is evaluated at its optimal operating condition while satisfying performance constraints.

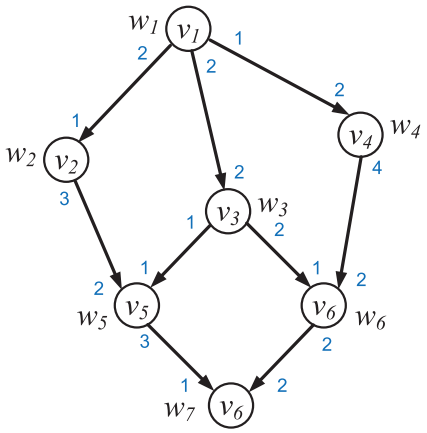

Fig. 3. A task graph representing a flow.

3. NBTI-AWARE WORKLOAD BALANCING

This section introduces the new workload balancing framework under NBTI impact. The proposed method groups cores into zones based on capacity rates. Each zone has one task flow. Task scheduling within the zone is formulated as a mixed-integer program (MIP) considering workload balancing and communication cost. Capacity rate is also incorporated in the scheduling model as a constraint of workload assignment. Capacity rates are frequently updated to accurately capture the time-dependent V_{th} degradation.

3.1. Dynamic Zoning

We propose Dynamic Zoning (DZ) to spread out the workload across the entire multi-core network. Workloads are fundamentally composed of sets of tasks, named task flows. Tasks within a particular flow may have dependencies among them, while tasks from different flows tend to have very few or no dependencies among them [El-Rewini et al. 1994]. Therefore, to minimize communication cost, we limit the process of one task flow to a group of cores physically adjacent to each other. Such a group of cores is defined as a *zone*. The DZ algorithm aims at mapping a task flow onto one particular zone. We then schedule tasks belonging to this flow among the cores within the assigned zone.

Given a task flow presented as a DAG (Directed Acyclic Graph) $G = (V, E)$ in Figure 3, nodes $V = \{v_1, v_2, \ldots, v_n\}$ represent a set of tasks to be executed. And the arcs $E = \{(i, j)\}$ specify precedence relationships: each arc (i, j) means that task v_i must be completed before v_j can start execution. Task weight w_i's represent the execution time of task v_i on a nonstressed core (i.e., a core with capacity rate 1). The numbers at the input and output of each node are the number of data tokens which represent the number of data packets are consumed (at input of node) or produced (at the output of each node) on each arc [Lee and Messerschmitt 1987]. Further details on the data tokens will be discussed in Section 3.2. The workload of this task flow can be evaluated by the structure of the task graph and the task weights. These two factors provide sufficient information for workload estimation. Each zone, once generated for a particular flow, should be able to accept the workload evaluated based on the given flow. On the other hand, as capacity rate specifies the upper bound of workload one core can accept, the sum of core capacity rates in one zone reflects the maximum total workload one zone can accept. This summation must exceed the estimated workload resulted from the flow assigned to this zone. We now explain how to evaluate the workload induced by a

particular flow. We sum up all the task weights of its task graph, and average the total sum over the length of the worst-case path because the worst-case path determines the longest execution time. The estimated workload induced by this task flow is defined as follows:

$$\text{Workload} = \frac{\sum_i w_i,\ \forall v_i \in V}{\sum_j w_j,\ \forall v_j \in WCP}, \tag{17}$$

where WCP denotes the set of nodes existing on the worst-case path. The workload estimation defined in (17) is the minimally required capacity rate for a required zone to process this particular flow, since capacity rate indicates the upper bound of acceptable assigned workload.

Zones are not restricted to rectangle shapes in the current article. Several factors determine the shape: capacity rates of included cores, total communication distances, and the size of zones (defined as the number of cores in each zone). We start zoning in a greedy way: start with a rectangular region for each zone and perturb gradually to meet the communication and capacity rate requirement. The requirement of capacity rate in this zone has been discussed above. To ensure low communication cost, a good zoning should have short Manhattan distances among cores in one zone. Thus, the problem of organizing an optimal zone for a particular flow can be described as:

$$\begin{aligned} \text{find} \quad & Z_k \\ \text{minimize} \quad & \sum_{(i,j)} d(i,j),\ \forall v_i, v_j \in Z_k \\ \text{subject to} \quad & \sum_i CR_i \geq \text{Workload},\ \forall v_i \in Z_k, \end{aligned} \tag{18}$$

where Z_k is the zoning result consisting of an optimal set of adjacent cores, $d(i, j)$ denotes the Manhattan distance between any two cores in this zone, and CR_i represents the capacity rate for each included core. The workload information of this flow is evaluated according to (17).

Algorithm 1 describes the proposed DZ method. When a new task flow comes into the system, DZ algorithm first searches for the maximally contiguous empty region of available cores ("Find_Max_Region"). Then starting from one corner of the explored empty region, "Initial_Rectangle" initializes a zone in rectangular shape. Rectangular shape leads to short Manhattan distances, and therefore is expected to be a good initial solution. "Manhattan_Distances" calculates the total communication distances in this zone. Then "Perturbation" makes slight adjustments to this initialized zone to explore a better grouping solution, according to a heuristic procedure described in Algorithm 1. For each adjustment, the heuristic compares the total distances between the initialized zone and the current adjusted zone. The heuristic not only accepts changes that improve the objective, but also some changes that deteriorate it. The latter are accepted probabilistically following a simulated annealing algorithm. This searching procedure is repeated until the termination condition is satisfied (the total distances of current zone is less than a predetermined threshold).

3.2. Task Scheduling in One Zone

As mentioned previously, the execution of tasks belonging to the same flow is restricted within one zone, because of no inter-zone dependency. After zoning, the subsequent step is to map the tasks onto the cores within this zone. We formulate the Dynamic Task Scheduling (DTS) problem as a mixed-integer program (MIP). The objective is to achieve maximum system utilization with minimum communication cost under workload constraints. A mixed-integer program is an optimization model in which some of

ALGORITHM 1: Dynamic Zoning

Input: a task flow to be allocated; a multi-core system in which each core's capacity rate is identified

Output: the optimal zoning result to execute the given flow

```
1: S ⇐ Find_Max_Region( );            ▷ comment: determine the maximally continuous region
2: DZ_opt ⇐ Initial_Rectangle(S);
3: Dist_opt ⇐ Manhattan_Distance(DZ_opt);   ▷ comment: compute the Manhattan distance for the optimal zoning result
4: k ⇐ 0;
5: while k < N and Dist_opt > D_th do
6:     DZ_new ⇐ Perturbation(DZ_opt);   ▷ comment: assign slight adjustments to explore better zoning result with shorter Manhattan distance
7:     Dist_new ⇐ Manhattan_Distance(DZ_new);
8:     if Dist_new < Dist_opt then   ▷ comment: update the optimal solution if a better zoning result is explored
9:         DZ_opt ⇐ DZ_new;
10:        Dist_opt ⇐ Dist_new;
11:    end if
12:    ΔDist ⇐ Dist_new – Dist_opt;
13:    if exp(–ΔDist / N) > random(0,1) then   ▷ comment: probabilistically accept some adjustments that deteriorate the objective
14:        DZ_opt ⇐ DZ_new;
15:        Dist_opt ⇐ Dist_new;
16:    end if
17:    k ⇐ k+1;
18: end while
19: return DZ_opt;
```

the decision variables (not all of them) have to be of type integer or binary [Wolsey and Nemhauser 1999].

To formulate the DTS problem into MIP form, we introduce a binary matrix M to represent all task-core mapping relationships. Let $V = \{v_1, v_2, \ldots, v_n\}$ denote a set of tasks to be executed within a zone of m cores. The task-core mapping matrix is thus of size $m \times n$. Each entry in this mapping matrix M_{ij} is a decision variable of binary type, which is set to "1" if task v_i is mapped onto the jth core and "0" as the opposite situation. The definition of this matrix obviously imposes the first set of constraints:

$$\sum_{i=1}^{m} M_{ij} = 1, \text{ for } j = 1, 2, \ldots, n., \tag{19}$$

which indicates that one task can be assigned to only one core. Using this binary mapping matrix, we conveniently determine task assignments.

Other than the mapping matrix, another set of decision variables are the starting times for all tasks, denoted by S_i's. Task starting times are combined with the task-core mapping matrix to determine the job sequences of all involved cores. An optimal schedule can be fully specified by determining the mapping matrix M_{ij}'s and task starting times S_i's: each row in the mapping matrix indicates a set of tasks to be executed on a particular core; the starting times help determine the execution sequence of all assigned tasks. We then present the second set of constraints in this optimization model, which is for the purpose of keeping the precedence relationships among tasks. For each arc (i, j) of the task graph, the precedence relationship requires that task i must be finished before the execution of its successor j:

$$S_i + T_i + T_{\text{comm}}(i, j) \leq S_j, \tag{20}$$

where S_i and T_i represent the starting time and execution time for task v_i, respectively, while T_{comm} represents the communication time between task i and j.

The third set of constraints are workload constraints imposed by core capacity rates. Capacity rate is included as an upper bound limit for workload assigned to each core. A stressed core with low capacity rate indicates that light workload should be assigned to this core. The proposed DTS method dynamically scales down core operating frequency and thus leads to varying task execution times. T_i for task i depends on which core this task resides in. Therefore, we need to take into account the scaling ratio when evaluating the assigned workload on a particular core. Let α_{ij} denote the frequency scaling ratio when core i is processing task j. For each core within the zone, its assigned workload can be estimated as:

$$WL_i = \frac{\sum_j \alpha_{ij} T_j \cdot M_{ij}}{T_s} \leq CR_i, \quad \text{for } i = 1, \ldots, m. \tag{21}$$

WL_i in (21) records the assigned workload on ith core. T_s represents the length of schedule, that is, the execution time on the worst-case path. The value of core's workload lies into the interval $[0, 1]$, and therefore can be compared with its capacity rate. The workload assigned to each core should not exceed its capacity rate. As defined previously, one core's capacity rate is measured based upon the optimal operating condition under power and timing constraints. The real-time operating condition is restricted to be worse than the optimal case to satisfy the performance constraints. Accordingly, when performing task scheduling the real workload assigned to this core cannot exceed the pre-evaluated bounding value; otherwise, it may break the performance limits and worsen the stress level.

We now discuss the objective functions in the proposed DTS method. For an efficient system, it is important to achieve high utilization. *Core Utilization* is measured as the ratio of its busy time to its total active period of time:

$$U_i = \frac{\sum_j (M_{ij} \cdot T_j)}{T_s}, \quad \text{for } i = 1, \ldots, m. \tag{22}$$

One goal in our optimization model is to optimize the overall system utilization summed over all cores. Under core workload constraints imposed by capacity rates, this total utilization is also bounded by total capacity rate of the zone.

Another goal in this DTS scheme is to minimize the total communication cost among the cores. In a task graph with different rates of data production and consumption [Lee and Messerschmitt 1987], some data tokens have to be stored in the buffer on the arc. Therefore, the communication cost consists of two components:

$$T_{\text{comm}} = T_{\text{trans}} + T_{\text{buff}}, \tag{23}$$

where T_{trans} denotes total transmitting cost, and T_{buff} denotes the total buffering cost (or storage cost). For each arc (i, j) in the task graph, the transmitting cost is computed as the multiplication of the number of token transmitted and the unit time to transmit one token. The total transmitting cost is summed up over all arcs with buffering operations:

$$T_{\text{trans}} = \sum_{(i,j)} N_c(i, j)\, c(i, j), \tag{24}$$

where $N_c(i, j)$ is the number of tokens transmitted on arc (i, j), and $c(i, j)$ is the unit transmission cost on (i, j). The unit transmission cost is the communication cost to transmit one data token on arc (i, j), which is dependent on the Manhattan distances

between the cores where the two tasks reside in:

$$c(i,j) \propto \sum_{(k,l)} M_{ik} \cdot M_{jl} \cdot \mathrm{dist}(k,l), \tag{25}$$

where $\mathrm{dist}(k,l)$ is the Manhattan distance between core k and core j. The buffering cost is calculated and accumulated in a similar way:

$$T_{\mathrm{buff}} = \sum_{(i,j)} N_b(i,j)\, b(i,j), \tag{26}$$

where $N_b(i,j)$ denotes the number of tokens to be buffered on arc (i,j) or loaded from (i,j), and $b(i,j)$ is the unit buffering cost. In case that the number of input data tokens is greater than that of output data tokens, $b(i,j)$ indicates the time it takes to buffer one unit token on (i,j). Under the opposite condition, $b(i,j)$ denotes the time it takes to load one token from the buffer on arc (i,j).

In summary, the DTS problem in our workload balancing framework can be generalized by the following mixed-integer program:

$$\begin{aligned}
\text{minimize} \quad & \alpha \cdot \left(\sum_{i=1}^{m} (1 - U_i) \right) + \beta \cdot T_{\mathrm{comm}} \\
\text{subject to} \quad & \sum_{i=1}^{m} M_{ij} = 1, \ \forall j = 1, 2, \ldots, n \\
& S_i + T_i + T_{\mathrm{comm}}(i,j) \le S_j, \ \forall (i,j) \subset E \\
& WL_i(M,S) \le CR_i, \ \forall i = 1, 2, \ldots, m \\
& M(i,j) \in \mathbf{Z}, \forall i = 1, 2, \ldots, m, \forall j = 1, 2, \ldots, n \\
& 0 \le M(i,j) \le 1, \forall (i,j) \\
\text{variables} \quad & M = \left[M_{ij} \right]_{m \times n}, \ S = \{S_1, S_2, \ldots, S_n\},
\end{aligned} \tag{27}$$

where U_i represents the recorded utilization of ith core, T_{comm} calculates the total communication cost based on the schedule. α and β are simply two weighting factors for the tradeoff between two objective functions. The decision variables are the task-core mapping matrix, with each entry M_{ij} constrained to be a binary value, as well as the task starting times. The mapping matrix determines all mapping relationships between tasks and corresponding assigned cores. Task starting times have to satisfy the precedence relationships. On each arc of the task graph, the source node must be finished before the execution of the sink node. Therefore, the core where the source node resides in has to operate at a certain speed to meet this timing deadline. With the mapping matrix and starting times specified, we are able to determine the frequencies for all cores responsible for task execution.

The first constraint in this optimization model is to guarantee the uniqueness of task-core mapping relationship. The second constraint satisfies all the precedence relationships among the tasks. The third constraint reflects the workload constraint induced by NBTI introduced capacity rate, where WL_i denotes the assigned workload on ith core, which is dependent on the mapping relationships and task starting times (refer to (21)), and CR_i represents its corresponding capacity rate.

Although the optimal solution can be derived by solving the MIP formulas in (27), its computational complexity is a serious problem when applied to realistic task graphs. It is well known that MIP-solving is often NP-hard [Schrijver 2003]. MIP solvers use the branch and bound algorithm [Schrijver 1998] to obtain possible values of integer variables. In this DTS algorithm, the size of the search space is proportional

to the number of combination patterns of integer variables, and therefore increases exponentially as number of tasks and cores increases. We use an integer heuristic-based algorithm to solve the MIP formulated in (27). The heuristic is based on the Local Branching approach [Fischetti and Lodi 2003], which aims at forming a new feasible solution of better objective value based upon one or more explored feasible solutions. The heuristic defines a neighborhood of a certain feasible solution, determines a point in this neighborhood which is optimal for the objective function. Such explored point is then used as a new reference point in the next iteration. The neighborhood of a feasible point is defined in terms of the Manhattan distance between two points. In such neighborhood we form a sub-optimization problem by exerting additional constraints to the original MIP, and the search procedure explores a new feasible solution by solving this sub-optimization problem. The details about this procedure can be found in Fischetti and Lodi [2003]. The search heuristic relies on the observation that the neighborhood of a feasible MIP solution often contains potentially better solutions.

3.3. NBTI-Aware Workload Balancing

This section details how to dynamically spread the workload across the entire network based on the proposed DZ and DTS methods. Each task flow is assigned to a particular zone for execution. The dynamic workload balancing policy should take effect in case of inserting or relaxing a particular zone. Moreover, the balancing strategy is adaptive to the frequent update of core capacity rate. As explained in Section 2, core capacity rate varies at different time points. In most situations, some of the cores may be in "quasi-defect" situations as they are over-stressed. The over-stressed cores cannot be assigned heavy workload at that moment. On the other hand, when these cores are released at a later time, they may become available again. In this sense, the capacity rate for a core is not a constant number but has to be updated frequently.

When a new flow comes into the system, the DZ algorithm takes effect immediately to generate an appropriate zone to allocate the flow tasks. The DZ algorithm first searches for available cores and explores the maximally contiguous region. Starting from the bottom-left corner of the region, the DZ algorithm determines the optimal grouping solution according to the heuristic described in Algorithm 1. After the cores are grouped in a zone, the DTS algorithm is responsible for mapping the tasks onto particular cores within this zone. Note that due to the generation of a new zone, the maximally contiguous region will then be updated. The procedure of generating a new zone is illustrated in Figure 4. The blocks in white color represent the cores having high capacity rates. These cores are not stressed, and therefore can accept heavy workload when performing task scheduling. The gray color shows that a core is in stressed status and can only be assigned light workload. The blocks in orange color represent the cores in the moderate status. Suppose a newly generated flow G_1 comes into the system, by employing DZ algorithm zone Z_1 is explored to process this task flow (denoted in blue color). At the same time, the maximally contiguous region has been changed. The cores in zone Z_1 should be excluded from core grouping before they are relaxed. After a certain period, another flow G_2 arrives, a new group of cores are organized to form a new zone Z_2 (denoted in yellow color) to process this flow. Apparently an update of the maximally contiguous region is required to reflect this zoning result.

The case of relaxing a zone is relatively simple. When a zone finishes processing all assigned tasks, all the cores within this zone will be relaxed to join other available cores for the processing of new flows. Therefore, the maximally contiguous region will be changed in next search iteration. This is done by a merging procedure. Moreover, after a period of task execution, the capacity rates of all involved cores also will be updated. Figure 5 illustrates the relaxation procedure of an existing zone. Suppose that zone Z_2 has finished processing flow G_2 earlier than zone Z_1 has, the five cores in

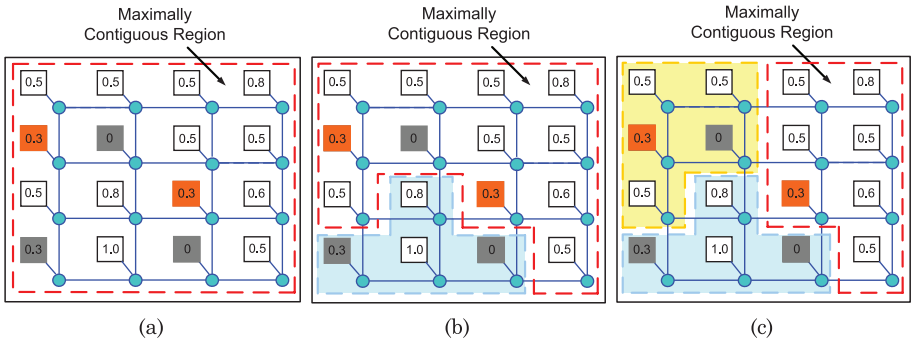

Fig. 4. The generation of a new zone.

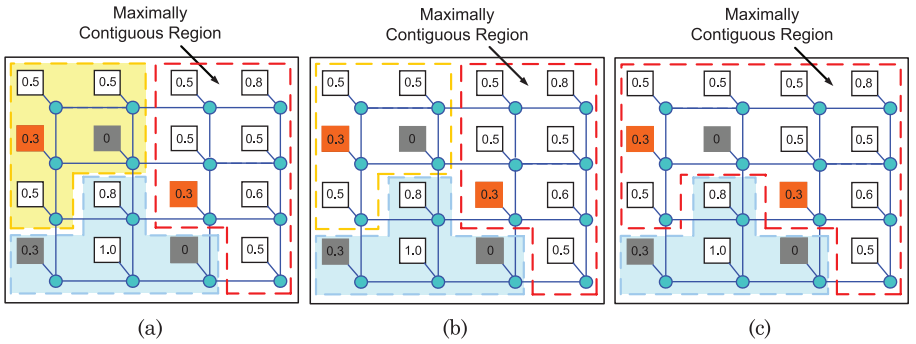

Fig. 5. The relaxation of an existing zone.

Z_2 will consequently be released and become available again. Such a relaxation of core zone results in a corresponding change in the maximally contiguous region. We also observe that the capacity rates of the cores in Z_2 have been changed after processing the assigned task flow.

The update of capacity rate depends not only on V_{th} degradation but also on workload assignment. As a consequence, capacity rate needs to be updated in two situations. On one hand, whenever a newly generated task graph arrives at the system, the dynamic task scheduling algorithm takes effect immediately to update capacity rate and assign the tasks onto the cores. On the other hand, capacity rate is updated on a periodic basis. The update period is determined according to an estimation of temperature shift trend. We use a thermal modeling tool HotSpot [Skadron et al. 2004] to predict the shortest time period to reach a pre-set temperature threshold (e.g., 85°C) for a core operating at full rate and full workload. This worst-case time estimate will be used as the update period of capacity rate. It is worth emphasizing that temperature is not explicitly shown in the optimization model because the evaluation of capacity rate has already taken into consideration of core's temperature shift. We rely on the temperature data obtained from HotSpot to predict the change of core's run-time temperature. Given core's operating speed and initial temperature as inputs, HotSpot is able to predict core's temperature change and power consumption after each calling interval.

4. EXPERIMENTAL RESULTS

This section presents the results of the proposed workload balancing methodology. We consider a 8×8 multicore system. We select an upper bound as the maximum number of cores that may be stressed due to NBTI aging effect. We initially started

Table I. Benchmark Applications Used for Profiling and the Generated Task Graphs

Application Name	Number of Task Nodes	Number of Arcs	Total Weights	Data Tokens
bc	12	23	14384	691
cjpeg	22	41	1128	1140
djkstra	25	42	523	1143
djpeg	21	38	3757	1058
fft	25	40	326	1109
qsort	24	44	1463	1201
rawcaudio	6	9	4829	231
ss	9	15	3637	445
susan	22	34	5021	956
tiff2bw	25	39	694	1140
tiff2rgba	25	48	847	1274
epicUnoptimizedEncode	14	26	4890	680
g721Decode	14	23	351	614
mpegDecode	18	36	5560	926
mpegEncode	23	35	4260	982
pegwitdecode	24	41	5464	1163
pegwitencode	20	34	5040	973
crcNetbench	10	19	3371	581
dhNetbench	25	47	4179	1322
drrNetbench	21	33	1337	1037
md5Netbench	25	44	4786	1262
tlNetbench	21	36	1285	1002

with a maximum number of eight cores and further increase this number to evaluate performance degradation and reliability improvement. As we can slow down each core by scaling down its operating frequency, NBTI would be less severe in these situations, which is beneficial for easing the stress level and improving system reliability.

To demonstrate the effectiveness of our proposed approach, all task graphs are generated by profiling realistic applications from several benchmark suites, including MediaBench [Lee et al. 2008], MiBench [Guthaus et al. 2001], and NetBench [Memik et al. 2001]. In the analysis to create task graphs, we selected individual functions (with at least 0.5% of the overall execution time) as the corresponding task nodes, from which we can create the task graphs by adding precedence relationships to them. We utilized the relative execution time for each function to compute the task weight for each task. In this way, the smallest task has a weight of 1, and all other tasks are presented in a multiple of this time unit. This implies that the computational weights across various task graphs would not reflect the varying degrees of complexity of those benchmark applications. The characteristics of the generated task graphs are summarized in Table I. The first column lists the names of all benchmark applications. The other columns provide the number of task nodes, the number of arcs, the total task weight, and the total number of data tokens for each task graph.

The proposed workload balancing framework is application-driven. Depending on the precedence relationships defined by the task flows, we partition the task flows considering workload under NBTI impact. During this process, we regard the multicore network as a whole system. The whole system is required to finish applications/tasks within certain user-specified time requirements. Individual cores may pause for certain amount of time and enter idle status. Table II lists the scheduling results by applying the proposed strategy at 90 seconds after the start of simulation. This table includes

Table II. The Capacity Rate and Core Utilization for NBTI Stressed Cores at 90s

NBTI Stressed Core Index	Zone # Grouped in	Capacity Rate	Assigned Workload
9	1	0.8193	0.6248
12	2	0.5923	0.5000
28	3	0.4550	0.3892
33	4	0.5012	0.4000
37	5	0.4242	0.4000
39	6	0.4979	0.4473
50	6	0.8238	0.6482
57	4	0.8932	0.8079

Table III. The Capacity Rate and Core Utilization for NBTI Stressed Cores at 100s

NBTI Stressed Core Index	Zone # Grouped in	Capacity Rate	Assigned Workload
14	2	0.3959	0.3514
25	2	0.4497	0.4000
31	3	0.5470	0.5000
38	4	0.5472	0.3892
41	4	0.7224	0.5864
44	6	0.3598	0.3000
52	5	0.6239	0.5536
58	6	0.5543	0.4293
59	6	0.2163	0.2000

core index, zone number each core is grouped in, each core's capacity rate and its assigned workload. Note that only the cores stressed by workload are listed in Table II. In addition, the scheduling results at 100 seconds are further presented in Table III. We can observe that for each NBTI stressed core, its assigned workload is well bounded by its capacity rate. The tables show that those cores stressed by workload at 90 seconds have been relaxed at 100 seconds, while a new group of cores alternate into stressed phase. The experimental results demonstrate that capacity rate is an indication of upper bound limit one core can accept workload. The DZ algorithm together with DTS algorithm adaptively manipulate the workload assigned to those NBTI stressed cores, in order to ease their stress levels.

We first evaluate network performance considering NBTI degradation. We use BookSim, a cycle-accurate interconnection network simulator, to evaluate system throughput. We initially ran simulations at a normal load up to 10000 sample periods, then reduce the load by 50% every 10000 periods. We terminate generation of new flows at 30000 sample periods as throughput statistics become stable. As shown in Figure 6, "Non-Stressed" represents the simulation without considering cores stress due to NBTI aging, while "Stressed" is the opposite case. Figure 6 shows that the throughput is almost identical before and after considering NBTI impact. In experiment, while the load on these 8/64 NBTI stressed nodes is reduced, other nodes are still at full rate. A 4% drop in throughput is observed in "Stressed" case. When workload increases beyond 0.3, minor differences can be observed between the ideal "Non-Stressed" and the "Stressed" cases. As the workload approaches the throughput limit, we observe that this difference saturates. This happens due to the fact that most of the packets are injected early in the network simulation, therefore no additional packets can be injected successfully into the system beyond saturation.

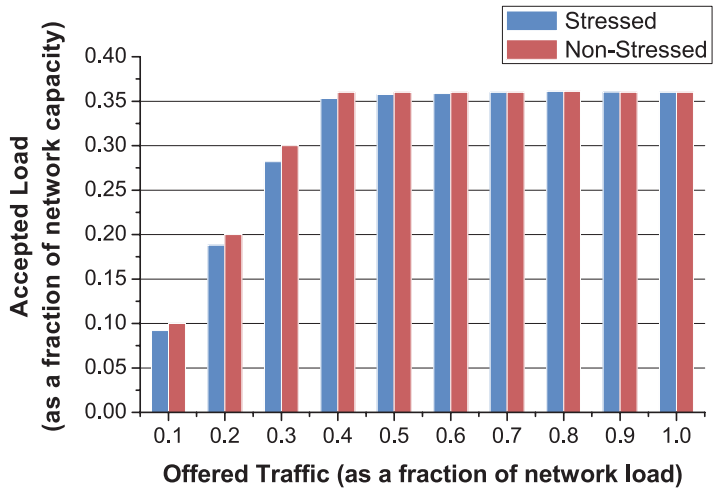

Fig. 6. Comparison of throughput between nonstressed and NBTI-stressed cases.

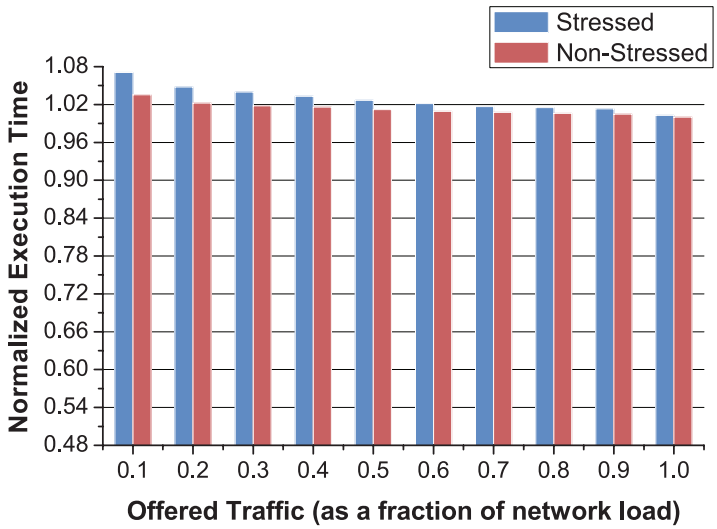

Fig. 7. Comparison of execution times between nonstressed and NBTI-stressed cases.

We run another set of experiments to compare system performance in terms of execution time. We observe how long the task flows run in such a 8×8 multicore system. In the same way, while the load on NBTI-stressed nodes is reduced, other nodes can operate at the maximum rate. Each NBTI-stressed core is able to execute these tasks at a certain frequency, which is associated with its capacity rate. We distribute the tasks by using the proposed DZ algorithm and evaluate the execution time by using the DTS algorithm. Figure 7 illustrates the overall task execution times (normalized to the metric on a system operating at full rate and full workload) at different levels of offered network load. Compared with the "Non-Stressed" case, an increase of less than 2% in execution time can be observed when the offered load is above 0.2. When offered load is beyond 0.5, less than 1% differences can be observed between the ideal "Non-stressed" and "Stressed" cases. For the worst case

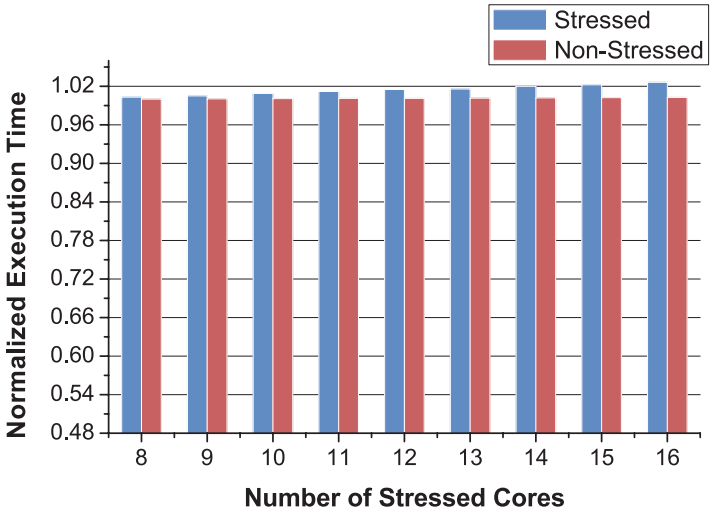

Fig. 8. Comparison of execution times with different number of NBTI-stressed cores.

that offered load as 0.1, approximately a 3% increase in execution time is obtained. The results show that the proposed strategy effectively balances the workload among NBTI-stressed cores, and therefore reduces system degradation to the most extent.

To demonstrate the efficiency of our proposed methodology, we increase the number of NBTI-stressed cores and investigate how system performance will be affected. Figure 8 shows the comparison of task execution time between "Stressed" and "Non-Stressed" cases. Here the offered load is fixed at full rate. Starting with 8 NBTI-stressed cores, the execution time is almost identical before and after applying the NBTI stress model. As the number of NBTI-stressed cores increases, a slight increase in the overall execution time can be observed. When the number of NBTI-stressed cores grows to 16, approximately a 3% performance drop is observed. The results demonstrate the efficiency of the method in balancing workload and alleviating aging effect on the devices stressed due to NBTI.

Figure 9 displays core failure rate (in percentage) with regard to time. The x-axis represents the time in terms of years and y-axis represents the percentage of core failure. The red solid line represents the result of our new approach while the blue dash line represents the case without the new approach. The difference in terms of yield becomes obvious after 2 years and begins to widen. For example, after 6 years, the core failure rate without the new methodology reaches as high as about twice of the result by using the proposed method. Furthermore, we used Monte-Carlo simulations to monitor the critical path delay and total leakage power for each core, and predicted the changes in MTTF. The MTTF estimation follows the derivation from Greskamp et al. [2007], Srinivasan et al. [2004, 2005], and Waldshmidt et al. [2006], where MTTF is modeled as an exponential function of operating temperature T. For temperature simulation during performance evaluation, we use HotSpot [Skadron et al. 2004], a popular thermal modeling tool, and rely on the temperature data obtained from HotSpot to predict the change of core's run-time temperature. Given core's operating speed and initial temperature as inputs, HotSpot is capable of predicting core's temperature shift and power consumption after each calling interval. To accurately characterize the thermal changes, HotSpot has to be initialized with a heat sink temperature for every simulation. For this purpose, we run HotSpot simulation twice with the first run to obtain core's average power consumption which

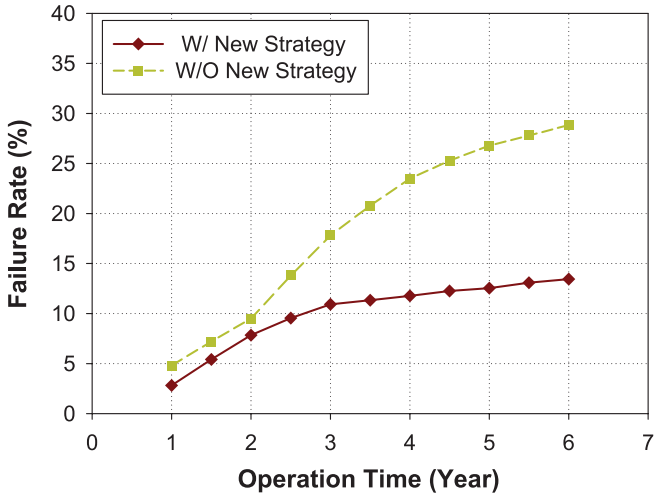

Fig. 9. Core failure percentage comparison between new strategy and without new strategy.

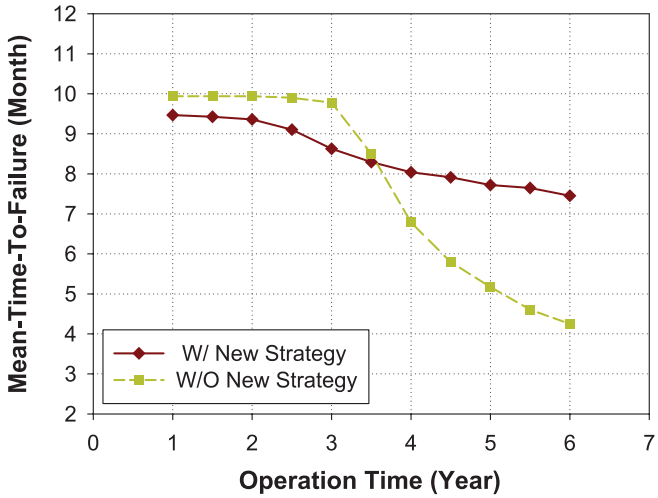

Fig. 10. MTTF comparison between new strategy and without new strategy.

can be used for heat sink temperature initialization. Figure 10 shows the MTTF comparison between multicore systems without the proposed methodology and with the proposed methodology. Considering the performance degradation brought by the proposed scheme, we have forced the system using the new strategy to operate at about 3% higher rate, along with the associated effects on temperature and reliability etc., to conduct an "apples-to-apples" comparison. The x-axis presents the time in terms of years of operation. The y-axis measure provides the average MTTF of the 8 multicore system with 8 NBTI stressed cores. Though after about 3 years, both cases observe decreases in MTTF measurements. The results indicate that by allowing relatively light workload assigned to the NBTI-stressed devices, the new strategy is beneficial to the recovery of NBTI stress and therefore extends device lifespan. On average, the new methodology demonstrates about 30% improvement in MTTF degradation.

5. CONCLUSIONS

This article presents a new design framework for multicore systems to include device wear-out impact. The new approach starts from device fractional NBTI model to evaluate core performance difference, and provides a new NBTI-aware system workload model based on new DZ and DTS algorithms to balance workload among active cores while relaxing NBTI stressed ones. Experimental results show that by allowing some cores stressed by workload to relax after a certain period of operation, the proposed methodology improves multicore system yield and extends system MTTF with graceful degradation in performance.

APPENDIX: MEAN AND VARIANCE OF DELAY VARIATION

Given a normal random variable X with mean value μ and standard deviation σ, consider a function of this random variable $Y = f(X)$. A Taylor series expansion around mean value μ yields:

$$Y = f(X) = f(\mu) + \sum_{n=0}^{\infty} \frac{f^{(n)}(\mu)}{n!} (X-\mu)^n . \tag{28}$$

The mean value of Y can be determined by taking the second-order expansion:

$$\begin{aligned} \mu_Y &= E\left[f(\mu) + f'(u)(X-\mu) + \frac{f''(u)}{2}(X-\mu)^2 \right] \\ &= f(\mu) + f''(u) \cdot \frac{\sigma^2}{2}. \end{aligned} \tag{29}$$

Recall the gate delay function (4), which is a function of V_{th} variation. Letting $V_x = V_{\text{dd}} - V_{\text{th}}$, the mean value can be obtained according to (29):

$$\begin{aligned} \mu_{D_g} &= \frac{V_{\text{dd}} L_{\text{eff}}}{\gamma} \cdot (V_x - \mu)^{-\alpha} + \frac{V_{\text{dd}} L_{\text{eff}}}{\gamma} \cdot (-\alpha)(-\alpha-1)(V_x-\mu)^{-\alpha-2} \cdot \frac{\sigma^2}{2} \\ &= \frac{V_{\text{dd}} L_{\text{eff}}}{\gamma} \cdot \left[(V_x-\mu)^{-\alpha} + (\alpha^2+\alpha)(V_x-\mu)^{-\alpha-2} \cdot \frac{\sigma^2}{2} \right]. \end{aligned} \tag{30}$$

To derive the variance of gate delay, we simply use the first-order expansion to estimate the function Y:

$$Y = f(X) = f(\mu) + f'(\mu)(X-\mu). \tag{31}$$

The mean value of Y is simply $\mu_Y = f(u)$. Therefore, the variance can be derived as:

$$\begin{aligned} \sigma_Y^2 &= E[(Y-\mu_Y)^2] = E\{[f'(u)(X-\mu)]^2\} \\ &= f'(u)^2 \cdot E[(X-\mu)^2] = f'(u)^2 \cdot \sigma^2. \end{aligned} \tag{32}$$

By substituting the gate delay function (4) into (32), the variance of gate delay can be derived as:

$$\begin{aligned} \sigma_{D_g}^2 &= \left(\frac{V_{\text{dd}} L_{\text{eff}}}{\gamma} \cdot (-\alpha)(V_x-\mu)^{-\alpha-1} \right)^2 \cdot \sigma^2 \\ &= \left(\frac{\alpha V_{\text{dd}} L_{\text{eff}}}{\gamma} \right)^2 \cdot (V_x-\mu)^{-2\alpha-2} \cdot \sigma^2. \end{aligned} \tag{33}$$

ACKNOWLEDGMENTS

The authors would like to thank Mr. Wesley Chu for helping run multicore network simulations, and collect required data for latency and throughput evaluations. The authors also would like to thank all the anonymous reviewers for their valuable suggestions to improve this article.

REFERENCES

ABELLA, J., VERA, X., AND GONZALEZ, A. 2007. Penelope: The NBTI-Aware Processor. In *Proceedings of International Symposium on Microarchitecture*. 85–96.

ALAM, M. AND MAHAPATRA, S. 2008. A comprehensive model of PMOS NBTI degradation. *Microelectron. Reliab. 45*, 1, 71–81.

BASOGLU, M., ORSHANSKY, M., AND EREZ, M. 2010. NBTI-aware DVFS: A new approach to saving energy and increasing processor lifetime. In *Proceedings of ISPLED*. 253–248.

BHARDWAJ, S., WANG, W., VTTIKONDA, R., CAO, Y., AND VRUDHULA, S. 2006. Predictive modeling of the NBTI effect for reliable design. In *Proceedings of CICC*. 189–192.

BILD, D., BOK, G., AND DICK, R. 2009. Minimization of NBTI performance degradation using internal node control. In *Proceedings of DATE*. 148–153.

CHEN, G., CHUAH, K. Y., LI, M. F., CHAN, D. S., ANG, C. H., ZHENG, J. Z., JIN, Y., AND KWONG, D. L. 2003. Dynamic NBTI of pmos transistors and its impact on device lifetime. In *Proceedings of IRPS*. 196–202.

CONSTANTINIDES, K., PLAZA, S., BLOME, J., BERTACCO, V., MAHLKE, S., AUSTIN, T., ZHANG, B., AND ORSHANSKY, M. 2007. Architecting a reliable CMP switch architecture. *ACM Trans. Architect. Code Optimizat. 4*, 1, 1–37.

COSKUN, A. K., ROSING, T. S., AND WHISNAN, K. 2007. Temperature Aware Task Scheduling in MPSoCs. In *Proceedings of DATE*. 1–6.

EL-REWINI, H., LEWIS, T. G., AND ALI, H. H. 1994. *Task Scheduling in Parallel and Distributed Systems*. Prentice Hall.

FISCHETTI, M. AND LODI, A. 2003. Local branching. *Math. Prog. 98*, 1–3, 23–47.

GRESKAMP, B., SARANGI, S. R., AND TORRELLAS, J. 2007. Threshold voltage variation effects on aging-related hard failure rates. In *Proceedings of ISCAS*. 1261–1264.

GUTHAUS, M. R., RINGENBERG, J. S., ERNST, D., AUSTIN, T. M., MUDGE, T., AND BROWN, R. B. 2001. MiBench: A free, commercially representative embedded benchmark suite. In *Proceedings of IEEE International Workshop on Workload Characterization*. 3–14.

HUNG, W.-L., XIE, Y., VIJAYKRISHNAN, N., KANDEMIR, M., AND IRWIN, M. J. 2005. Thermal-aware task allocation and scheduling for embedded systems. In *Proceedings of DATE*. 898–899.

LEE, C., POTKONJAK, M., AND MANGIONE-SMITH, W.-H. 2008. MediaBench: A tool for evaluating and synthesizing multimedia and communications systems. In *Proceedings of MICRO*. 330–335.

LEE, E.-A. AND MESSERSCHMITT, D. G. 1987. Synchronous data flow. *Proc. IEEE 75*, 9, 1235–1245.

LIN, C.-H., LIN, I.-C., AND LI, K.-H. 2011. TG-based technique for NBTI degradation and leakage optimization. In *Proceedings of ISPLED*. 133–138.

MEMIK, G., MANGIONE-SMITH, W. H., AND HU, W. 2001. NetBench: A benchmarking suite for network processors. In *Proceedings of ICCAD*. 39–42.

PAPOULIS, A. 2002. *Probability, Random Variables and Stochastic Processes*. McGraw-Hill, New York.

PAUL, B. C., KANG, K., KUFLUOGLU, H., ALAM, M. A., AND ROY, K. 2005. Impact of NBTI on the temporal performance degradation of digital circuits. *IEEE Electron Dev. Lett. 26*, 8, 560–562.

REDDY, V., KRISHNAN, A. T., MARSHALL, A., RODRIGUEZ, J., NATARAJAN, S., ROST, T., AND KRISHNAN, S. 2002. Impact of negative bias temperature instability on digital circuit reliability. In *Proceedings of IRPS*. 248–254.

RONG, P. AND PEDRAM, M. 2006. Power-aware scheduling and dynamic voltage setting for tasks running on a hard real-time system. In *Proceedings of ASPDAC*. 473–478.

RUGGIERO, M., GUERRI, A., BERTOZZI, D., POLETTI, F., AND MILANO, M. 2006. Communication aware allocation and scheduling framework for stream-oriented multi-processor system-on-chip. In *Proceedings of DATE*. 3–8.

SARANGI, S., GRESKAMP, B., TIWARI, A., AND TORRELLAS, J. 2008a. Eval: Utilizing processors with variation-induced timing errors. In *Proceedings of MICRO*. 423–434.

SARANGI, S. R., GRESKAMP, B., TEODORESCU, R., NAKANO, J., TIWARI, A., AND TORRELLAS, J. 2008b. VARIUS: A model of process variation and resulting timing errors for microarchitects. *IEEE Trans. Semi. Manu. 21*, 1, 3–13.

SCHRIJVER, A. 1998. *Theory of Linear and Integer Programming*. Wiley.

SCHRIJVER, A. 2003. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer.

Skadron, K., Stan, M. R., Sankaranarayanan, K., Huang, W., Velusamy, S., and Tarjan, D. 2004. Temperature-aware microarchitecture: modeling and implementation. *ACM Trans. Architect. Code Optim. 1*, 1, 94–125.

Srinivasan, J., Adve, S. V., Bose, P., and Rivers, J. A. 2004. The impact of technology scaling on lifetime reliability. In *Proceedings of Dependable Systems and Networks*. 177–186.

Srinivasan, J., Adve, S. V., Bose, P., and Rivers, J. A. 2005. Exploiting structural duplication for lifetime reliability enhancement. In *Proceedings of ISCA*. 520–531.

Sun, J., Ma, D., Li, J., and Wang, J. M. 2008. Chebyshev-affine-arithmetic based parametric yield prediction under limited descriptions of uncertainty. *IEEE Trans. Comput. Aid. Design Integ. Circ. Syst. 27*, 10, 1852–1866.

Waldshmidt, K., Haase, J., Hofmann, A., Damm, M., and Hauser, D. 2006. Reliability-aware power management of multi-core systems (MPSOCS). In *Proceedings of Dynamically Reconfigurable Architectures*. 520–531.

Wang, W., Yang, S., Bhardwaj, S., Wattikonda, R., Vrudhula, S., Liu, F., and Cao, Y. 2007a. The impact of NBTI on the performance of combinational and sequential circuits. In *Proceedings of DAC*.

Wang, Y., Luo, H., He, K., Luo, R., Yang, H., and Xie, Y. 2007b. Temperature-aware NBTI modeling and the impact of input vector control on performance degradation. In *Proceedings of DATE*. 546–551.

Wolsey, L. A. and Nemhauser, G. L. 1999. *Integer and Combinatorial Optimization*. Wiley-Interscience.

Zhang, S., Wason, V., and Banerjee, K. 2004. A probabilistic framework to estimate full-chip threshold leakage power distribution considering within-die and die-to-die P-T-V variations. In *Proceedings of ISLPED*. 156–161.

Received March 2012; revised July 2012; accepted November 2012