Communication and Migration Energy Aware Design Space Exploration for Multicore Systems with Intermittent Faults

Anup Das, Akash Kumar and Bharadwaj Veeravalli
Department of Electrical and Computer Engineering
National University of Singapore
Email: {akdas, akash, elebv}@nus.edu.sg

***Abstract*—Shrinking transistor geometries, aggressive voltage scaling and higher operating frequencies have negatively impacted the dependability of embedded multicore systems. Most existing research works on fault-tolerance have focused on transient and permanent faults of cores. Intermittent faults are a separate class of defects resulting from on-chip temperature, pressure and voltage variations and lasting for a few cycles to several seconds or more. Operations of cores impacted by intermittent faults are suspended during these cycles but come back alive when conditions become favorable.**

This paper proposes a technique to model the availability of multiprocessor systems-on-chip (MPSoCs) with intermittent and reparable device defects. This model is based on Markov chain with stochastic fault distribution and can be applied even for permanent faults. Based on this model, a design space pruning technique is proposed to select a set of task mappings (with variable resource usage), which minimizes the task communication energy while satisfying the MPSoC availability constraint. Moreover, task migration overhead is also minimized, which is an important consideration for frequently occurring intermittent and temperature related faults, where prolonged system downtime during task re-mapping is not desired. Experiments conducted with real-life and synthetic application task graphs demonstrate that the proposed technique minimizes communication energy by 30% and reduces migration overhead by 50% as compared to the existing approaches.

I. Introduction

Shrinking feature-size and growing transistor density are negatively impacting the dependability of multiprocessor systems-on-chip (MPSoCs) by increasing the chances of faults (permanent, intermittent and transient) [1]. Recently, intermittent device defects are gaining significant attention among research community [2]–[4]. These are a class of hardware faults occurring frequently but irregularly over a period of time due to process, voltage and temperature (PVT) variations. A recent study with error logs from 950,000 PCs reveals that 39% of the hardware errors are intermittent [4]. Beside MTTF, one important performance metric for MPSoC with intermittent faults is its availability i.e. the percentage of time it is available to execute an application.

Task re-mapping has shown significant promise to tolerate transient and permanent faults [5]–[13]. Intermittent faults present separate challenges for system designers due to the unpredictability associated with their occurrence and their duration. Naively suspending the system operation for the intermittent fault duration can degrade system performance significantly. Unlike transient faults, task re-execution cannot guarantee elimination of the intermittent faults as they usually persist for a few cycles, if not for a few seconds or more. Permanent fault-aware task-mapping techniques, those minimizing failure probabilities, are not effective for intermittent faults either. This is due to the non-consideration of migration overhead which can lead to significant system downtime making the system unavailable for the entire fault duration and a similar downtime after fault duration to restore the tasks back to their original cores. Thus, although permanent fault-aware mappings maximize the useful life of a system, the availability requirement is often violated as shown in Section VII.

Another research direction for embedded multiprocessor systems is concerning energy consumption. Researchers have shown that carefully selecting an application task mapping can significantly reduce task communication energy which constitutes a large fraction ($\approx 60\%$ according to [14]) of the overall energy consumption. This is orthogonal to dynamic voltage or frequency scaling capabilities of an MPSoC which can further reduce the task computation energy.

Contributions: Following are the key contributions.

- Markov modeling of multiprocessor system availability with stochastic fault-distribution.
- Task mapping technique to minimize the application communication energy while satisfying the MPSoC availability constraint.
- Minimization of task migration overhead for frequently occurring intermittent and temperature related faults.
- A heuristic to minimize the execution time of the energy, reliability and migration aware design space exploration.

A closed form expression for MPSoC availability is derived using Markov state transition. Based on this expression, analysis is performed at compile-time to obtain an initial task-mapping which minimizes communication energy and task migration overhead while satisfying the MPSoC availability constraint. Experiments with real-life and synthetic application task graphs on MPSoCs with homogeneous cores demonstrate that the proposed approach minimizes communication energy by 30% and the migration overhead by 50% as compared to the existing permanent fault-tolerant approaches. Moreover, the proposed heuristic reduces the design space exploration time by reducing mappings by more than 90% with satisfactory result quality (within 10% variation from minimum energy).

To the best of our knowledge, this is the first work which models MPSoC availability (using Markov chain) considering intermittent and reparable faults and integrate the same in application mapping.

The rest of the paper is organized as follows. A brief overview of the prior works in fault-tolerance and energy minimization domain is provided in Section II. This is then followed by preliminaries on MPSoC reliability and the design methodology in Sections III and IV respectively. The detailed problem formulation is provided next in Section V. Section VI introduces the readers to the proposed heuristic for solving the multi-criterion optimization problem. Finally, Section VII presents the results and different trade-off analyses and Section VIII concludes the paper with key future directions.

II. Related Works

A. Permanent Fault-Aware Task Mapping

Permanent fault aware task mapping techniques generate a starting application mapping with the aim of maximizing the lifetime (measured as *Mean Time To Failure (MTTF)*) of an MPSoC. Authors in [6]–[8] have shown that the failure rate during useful operating life of a core is constant and therefore exponential model can be safely assumed for the lifetime distribution. Core aging effects such as electromigration, stress-migration and thermal cycling are modeled into temperature and the same is incorporated in the application mapping and scheduling. There is a second category of research considering Weibull and Lognormal models for core lifetime distribution [9] [10]. Authors in these works considered factors such as temperature, operating frequency and voltage explicitly in the scale parameter of the distribution. There are three limitations of these works. First, mappings are generated based on a series-failure assumption i.e. an MPSoC fails when the first core becomes faulty (single fault consideration). This assumption is not true, in general, for modern MPSoCs, where tasks can be remapped after fault. Second, these works generate a starting mapping considering the fact that a core once faulty can never come alive. These mappings when applied to reparable and intermittent faults, can lead to sub-optimal results in terms of MPSoC availability, as shown in Section VII. Third, task communication energy and migration overhead are not considered in these works. There is one recent work incorporating multiple failures in task-mapping generation [11]. However, this work also suffers from the second and the third limitations discussed above.

B. Energy and Fault-Aware Task Mapping

Most of the prior attempts to integrate energy and fault-tolerance in the application mapping generation have focused on transient faults [15]. Recently, there is one work on generating task-mapping with the joint objective of reducing task communication energy and maximizing system MTTF (under permanent fault) [12]. However, consideration of reparable and intermittent faults, migration overhead and multiple core failure-based analysis are lacking.

To summarize, Table I provides the requirements for energy and reliability aware design space exploration for reparable and intermittent faults and limitations of the existing works.

TABLE I
Requirements and Limitations of Related Works

Related Works	Weibull Distribution	Multiple Core Failures	Reparable Faults	Comm. Energy	Migration Overhead
[6]–[8]	×	×	×	×	×
[9], [10]	√	×	×	×	×
[11]	√	√	×	×	×
[12]	√	×	×	√	×
Proposed	√	√	√	√	√

III. Preliminaries on Intermittent Faults

A. Causes of Intermittent Faults

There are three major factors contributing to intermittent faults – wear-outs, manufacturing and design defects. Wear-outs such as negative bias temperature instability (NBTI) and time dependent dielectric breakdown are susceptible to PVT variations at deep submicron nodes (45nm and beyond).

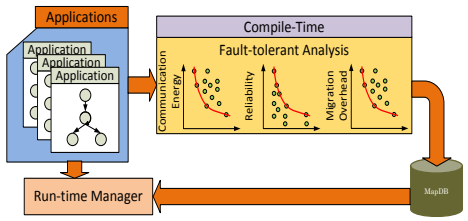

Fig. 1. Design Methodology

Another major cause for intermittent faults is manufacturing defects. Although, deterministic defects are detected in the testing phase, non-deterministic faults manifest as intermittent faults. Finally, design defects, those triggering from rare corner-cases can also lead to intermittent faults.

B. Intermittent Fault Occurrence and Distribution

The lifetime of a core (denoted by t) is a continuous random variable taking non-negative values. There are several distributions considered in literature for modeling faults in a core. This work assumes Weibull distribution with a base MTTF of 6.56 years to model the occurrence of intermittent burst errors in a core. However, during each error burst the fault is modeled using Poisson distribution [2].

The probability density function of Weibull distribution is

$$f(t) = \frac{\beta t^{\beta-1}}{\eta^{\beta}} e^{-(t/\eta)^{\beta}}, \quad t \geq 0 \tag{1}$$

where β is the shape parameter and η is the scale parameter. The reliability (R) and the failure rate (λ) are given by

$$R(t) = \int_{t}^{\infty} f(t)\,\mathrm{d}t \; = e^{-(t/\eta)^{\beta}} \text{ and } \lambda(t) = \frac{\beta}{\eta}(t/\eta)^{\beta-1}$$

There are three dominant wear-out effects studied for ICs: electromigration (EM), time-dependent dielectric breakdown (TDDB) and thermal cycling (TC). EM related wear-out failures are assumed here; however, any other effects can be easily incorporated by changing the scale parameter.

The scale parameter due to EM is calculated using Equation 2 (Black's equation ref [16]) where Γ is the gamma function, A_0 and n are material-related constant, $J(J_{crit})$ is the (critical) current density, E_a is the activation energy, K is the Boltzman's constant and T is the temperature.

$$\eta = \frac{MTTF(EM)}{\Gamma\left(1+\frac{1}{\beta}\right)} = \frac{A_0 (J - J_{crit})^{-n} e^{\frac{E_a}{KT}}}{\Gamma\left(1+\frac{1}{\beta}\right)} \tag{2}$$

IV. Design Flow

The task mapping methodology consists of two phases – analysis at compile-time and execution at run-time. The focus of this research is on the compile-time analysis; however, for the sake of completeness, a brief overview is provided on how to use the compile-time analysis results at run-time.

A. Compile-Time Analysis

The compile-time design methodology (highlighted in Figure 1) consists of determining an application mapping which minimizes the communication energy and migration overhead for every application enabled on the platform. The result is a set of mappings e.g. $\{M_n^{FFT}, M_{n-1}^{FFT}, \cdots, M_k^{FFT}\}$, where M_n^{FFT} denotes mapping of application *FFT* on the MPSoC

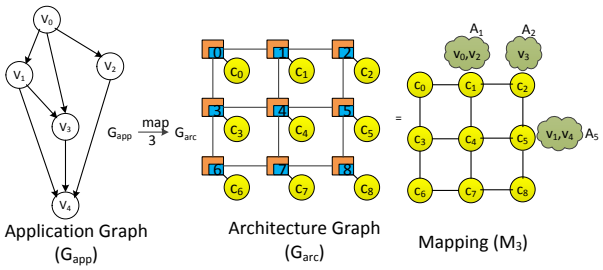

Fig. 2. Mesh-based MPSoC architecture

with n cores and k is the minimum number of cores required to satisfy the throughput requirement of *FFT*. Each mapping M_i^{FFT} in this set is selected based on three parameters – availability requirement (refer Section V-B), communication energy of M_i^{FFT} (refer Section V-D) and the migration overhead from M_{i+1}^{FFT} to M_i^{FFT} (refer Section V-E). The initial mapping (e.g. M_n^{FFT}) is generated according to [17]. When a single application is referenced, the superscript from the mapping notation is omitted for simplification (refer Figure 2).

B. Run-Time Resource Management

At run-time, a mapping is fetched from the mapping database depending on the application enabled and the number of cores available. When a core fails, the system fetches the next mapping from the application mapping set which minimizes the communication energy while incurring the minimum migration overhead. Thus, the platform downtime is minimized together with its energy consumption while continuing to be available as per the system specification.

V. Problem Formulation

A. Application and Architecture Model

An application is a directed graph $G_{app} = (V_{app}, E_{app})$, where V_{app} is the set of nodes representing tasks of the application and E_{app} is the set of edges representing data dependency among tasks. Each task $v_i \in V_{app}$ is a tuple $\langle T_i, S_i, \{D_{ij}\}\rangle$, where T_i is the execution time of v_i, S_i is its state space (program and data memory) and D_{ij} is the data produced on edge e_{ij} at every execution of v_i.

A mesh-based MPSoC architecture is assumed, similar to the one shown in Figure 2. An architecture is represented as a graph $G_{arc} = (V_{arc}, E_{arc})$, where V_{arc} is the set of nodes representing cores and E_{arc} is the set of edges representing communication channels among the cores. Mapping of an application G_{app} to architecture G_{arc} is represented as follows.

$$G_{app} \xrightarrow[n]{map} G_{arc} := G(V_{arc}, V_{app}) := M_n \tag{3}$$

where n is the number of cores of the MPSoC used by an application. With every core $c_i \in V_{arc}$, a set A_i is associated, consisting of the task(s) mapped to c_i. Thus, $V_{app} = \cup_{i=1}^{n} A_i$.

B. Availability Modeling of MPSoC

The steady-state availability of an MPSoC is modeled as a continuous-time discrete-state Markov Chain with the state of the system defining the number of available cores. Two repair techniques are considered.

- *Self Repair:* Intermittent temperature-related defects are self-healing. A faulty core comes alive once it cools down.

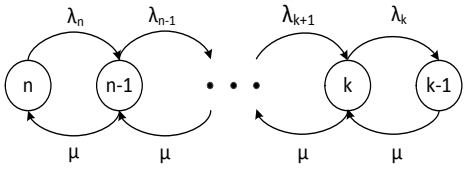

Fig. 3. State transition diagram

- *Dedicated Repair Mechanism:* In some of the modern MPSoCs, dedicated hardware module(s) are present to repair a faulty core.

*1) **Model Description and Assumptions**:*

- *Faults are self-revealing*
- *Faults on the active cores are independent*
- *Probability of more than one core failures in a small time interval Δt is negligible*
- *Application throughput requirement*
  An MPSoC fails in state i, if throughput of an application cannot be satisfied with i cores. For a graceful degrading system, an MPSoC can operate as long as atleast one core is fault-free albeit at a reduced throughput.

*2) **Notations Used in the Availability Expression**:*

i	active cores in the MPSoC, $k \leq i \leq n$
$\lambda_i(t)$	failure rate of the MPSoC with i active cores at time t
μ	constant repair rate of a core
$P_i(t)$	probability that there are i active cores at time t
$\hat{P}_i$	steady-state probability = $\lim_{t\to+\infty} P_i(t)$
A_s	steady-state availability of the MPSoC

The failure rate (λ_i) in state i is determined by the failure rate of the core (in state i) with the minimum MTTF. This is similar to the assumption in [6]–[12]. Detailed analysis with failure rates of all cores in a state is subject to future research.

*3) **Closed-from Expression for MPSoC Availability**:*
Based on the model description and the notations, the state transition diagram of the Markov model is shown in Figure 3. The state $(k-1)$ is an all absorbing state. The MPSoC fails when it reaches this state. The following events are defined.

- *S:* The MPSoC is at state i at time $(t+\Delta t)$.
- *S1:* The MPSoC is at state i at time t and no state transition occurred in the time interval Δt.
- *S2:* The MPSoC is at state $(i+1)$ at time t and a transition to state i occurs in interval Δt (failure).
- *S3:* The MPSoC is at state $(i-1)$ at time t and a transition to state i occurs in interval Δt (repair).

Clearly, $S = S1 \cup S2 \cup S3$.

The state probabilities are written as shown in Equation 4.

$$\begin{aligned} P_i(t+\Delta t) \quad = \quad & P_i(t)(1-\lambda_i\Delta t)(1-\mu\Delta t) + \\ & P_{i+1}(t)(1-\mu\Delta t)\lambda_{i+1}\Delta t + \\ & P_{i-1}(t)(1-\lambda_{i-1}\Delta t)\mu\Delta t \end{aligned} \tag{4}$$

where the boundary conditions are $P_n(0) = 1$ and $P_i(0) = 0$, for $k-1 \leq i \leq n-1$, i.e. all the cores are assumed to be operational at time $t = 0$. Taking $\lim_{\Delta t\to 0}$ on both sides

$$\begin{aligned} \frac{dP_i(t)}{dt} \quad &= \quad P_{i-1}(t)\mu + P_{i+1}(t)\lambda_{i+1} - (\lambda_i+\mu)P_i(t) \\ & \qquad k-1 \leq i \leq n-1 \\ \frac{dP_n(t)}{dt} \quad &= \quad -\lambda_n P_n(t) + \mu P_{n-1}(t) \end{aligned} \tag{5}$$

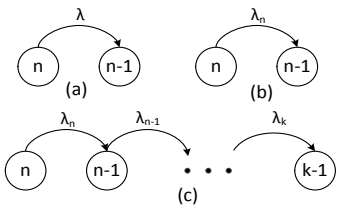

Fig. 4. Representation of prior works in terms of state diagram

It is assumed that an MPSoC operation is halted once it reaches state $(k-1)$. Therefore, the MPSoC must be in one of the $(n-k+2)$ states at any instance of time.

$$P_n(t) + P_{n-1}(t) + \cdots + P_k(t) + P_{k-1}(t) = 1 \tag{6}$$

The steady-state availability is calculated by evaluating both sides of Equation 5 and 6 with $\lim_{t\to+\infty}$ and using $\lim_{t\to+\infty} \frac{dP_i(t)}{dt} = 0$.

$$A_s = 1 - \hat{P}_{k-1} = \left[1 - \frac{\prod_{i=n}^{k} \lambda_i}{\sum_{i=n}^{k} \left(\mu^i \prod_{j=i+1}^{n} \lambda_j\right)}\right] \tag{7}$$

C. Extension of the Markov Model to Existing Research

Figure 4 shows the adaptation of the Markov model of Figure 3 to prior research. Figure 4(a) represents works [6]–[8] where the failure rate is constant. Research works of [9], [10], [12] can be described using Figure 4(b) where MPSoC analysis is based on a single core failure. Finally, the fault model of [11] is represented using Figure 4(c).

D. Communication Energy Modeling of Applications

Application communication energy is determined by

- the amount of data communicated over NoC links
- energy consumed in communicating a single bit of data
- the hop distance over which data is communicated

The energy per bit consumed in transferring data between core c_i and core c_j, situated $n_{hops}(i,j)$ away is given by

$$E_{bit}(i,j) = n_{hops}(i,j) \times E_{S_{bit}} + (n_{hops}(i,j) - 1) \times E_{L_{bit}}$$

where $E_{S_{bit}}$ and $E_{L_{bit}}$ are the energy consumed on the switch and the link respectively. $n_{hops}(i,j)$ is the number of routers between core c_i and c_j. The data communicated between these two cores is

$$data(i,j) = \sum_{\substack{\forall v_k \in a_i \\ \forall v_l \in a_j}} D_{kl} \times E_{k,l} \text{ where } E_{k,l} = \begin{cases} 1 & \text{if } (k,l) \in E \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

The total communication energy is therefore given by

$$E_{comm} = \sum_{\substack{\forall i,j \\ i \neq j}} data(i,j) \times E_{bit}(i,j) \tag{9}$$

Algorithm 1 Communication and migration energy optimization

Input: Application and architecture graph: G_{app}, G_{arc}; MPSoC availability constraint C
Output: A mapping that minimizes communication energy, satisfying the availability requirement

1: Determine k, the minimum number of cores needed for MPSoC to continue operation
2: ψ = mappings of G_{app} on G_{arc} with n cores sorted on communication energy
3: Initialize $i = 0$
4: **while** $A_s < C$ **do**
5: $\quad M_n = \psi[i++]$
6: $\quad \Xi$ = construct-m-ary-mapping-tree(M_n)
7: $\quad \Omega$ = Generate-root-to-leaf-path(Ξ)
8: $\quad \forall \omega \in \Omega, \;\; A.push(ComputeAvailability(\omega))$
9: $\quad A_s = getMax(A)$
10: **end while**
11: Return $\psi[i-1]$

E. Modeling of Migration Overhead

Migration overhead is an important consideration for the problem at hand. This is due to high overhead associated with re-mapping of tasks following a fault. This can lead to deadline misses or system downtime and is not desirable especially for intermittent faults due to its frequent occurrence. One of the optimization objectives is therefore to minimize the migration overhead which is computed as the cost incurred in moving from mapping of state l to state $(l-1)$ and back (refer Figure 3). For easier representation, the migration overhead is measured by migration energy as shown in Equation 10.

$$MC(l-1|l) \quad = \sum_{\forall v_u \in V_{app}} S_u \times E_{bit}(i,j) \qquad k \leq l \leq (n-1) \tag{10}$$

where c_i and c_j are respectively the core on which v_u is mapped in mapping M_l and M_{l-1} respectively. Clearly, minimizing the migration overhead is equivalent to minimizing the migration energy.

F. Multi-Criteria Optimization Problem

The table below summarizes the optimization problem.

Minimize $\sum_{l=k}^{n-1} E_{comm} \times MC(l|l-1)$
Subject to

- All tasks meet their respective deadlines
- All control/data dependencies are satisfied
- $A_s \geq$ MPSoC availability constraint
- MTTF $\geq$ MPSoC MTTF constraint

MPSoC MTTF is computed according to [13].

VI. Heuristic Solution Approach

Combining the migration overhead with availability and communication energy leads to a convex optimization objective. Although efficient tools exist (`cvx` for example) to solve this convex problem, the execution time is usually large and is not scalable with the number of tasks and cores. To reduce the execution time for the solution, a heuristic is proposed. This is shown as pseudo-code in Algorithm 1.

The first step of the algorithm is to determine the minimum number of cores (k) needed to satisfy application throughput requirement (line 1). A set of mappings is then generated. These mappings are sorted based on communication energy and stored in the set ψ.

TABLE II
REFERENCES AND THEIR ABBREVIATIONS FOR COMPARISON

References	Abbreviation	Description
[6]–[8]	MT	Multi-fault Temperature optimization
[9], [10]	SM	Single fault MTTF optimization
[11]	MM	Multi-fault MTTF optimization
[12]	SME	Single fault MTTF & Energy optimization
Proposed	MMEAR	Multi-fault MTTF, Energy & Availability optimization considering Repair

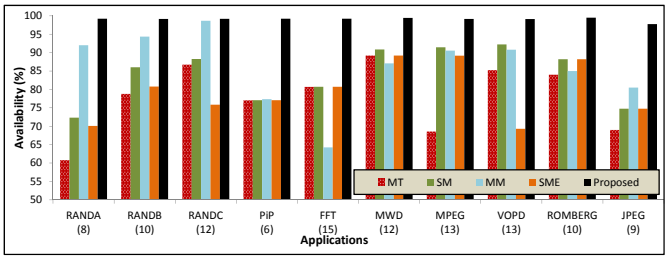

Fig. 5. Steady state availability

An m-ary mapping tree is generated with mapping M_i as root node (line 6). There are $(n-k)$ levels of the tree (corresponding to the states k to $(n-1)$ in Figure 3) and m branches spanning out from each node. A node at level l of the tree represents $G_{app} \xrightarrow[l]{map} G_{arc}$. Each branch b_{xy} connects source node x (mapping M_x) with destination node y (mapping M_y) and has weight equal to the product of migration overhead from M_x to M_y, the communication energy and the failure rate associated with M_y[1].

Once the tree is generated, all paths from root to the leaf nodes are identified (line 7). For each path, the availability of the MPSoC is determined by considering the failure rate associated with every node (mappings) of the path. If the availability requirement of the platform is met, the algorithm terminates; else, the local optimization step is re-executed starting with the next mapping from the set ψ.

VII. RESULTS AND ANALYSIS

The proposed algorithms are implemented in C++ and integrated with Matlab. The following parameters are used for the computations of failure rate [9]: current density $J = 1.5 \times 10^6 A/cm^2$, activation energy $E_a = 0.48eV$, the slope parameter $\beta = 2$, temperature $T = 350K$ and $n = 1.1$.

The results of the proposed approach are compared with references [6]–[12]. These are abbreviated as shown in Table II.

A. Steady State Availability

Experiments are conducted with 50 synthetic application task graphs with the number of tasks selected randomly from the range 4 to 32 and MPSoCs consisting of 2 to 8 homogeneous cores. Additionally, a set of real-life applications are considered both from streaming and non-streaming domain.

Figure 5 plots the steady state availability (in percentage) of all five techniques of Table II, for 10 applications randomly selected from the exhaustive set above. The number of tasks in each application is indicated in parenthesis against its name. These applications are executed on an MPSoC with 6 homogeneous cores (arranged as 2×3).

[1]It should be noted that although product function is used, the algorithm can be easily modified to incorporate any other function.

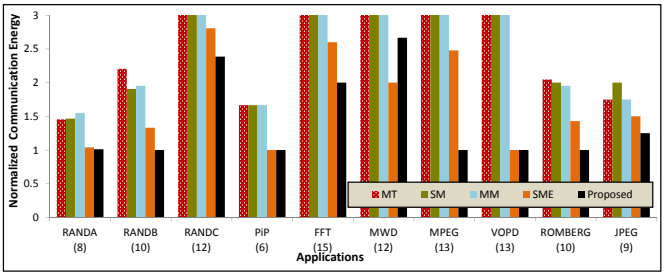

Fig. 6. Task communication energy

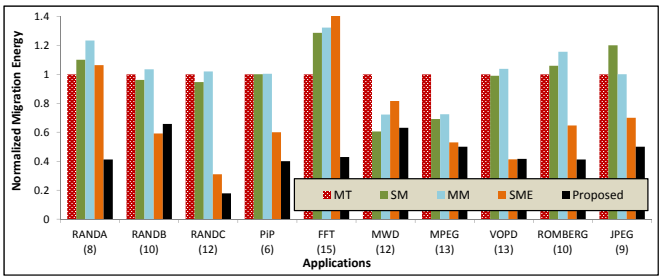

Fig. 7. Normalized task migration overhead

For each application in the figure, there are five bars, four of which are for the existing techniques (refer Table II). The availability requirement for the MPSoC is set to 99%. When the existing mapping techniques are applied on the MPSoC, the availability requirement is violated by 20% on average for most applications. The proposed technique satisfied the availability requirement for all 10 applications in Figure 5. Although not explicitly shown here, only 2 of the 50 synthetic applications violated the constraint by less than 0.05% with the design parameter, m set to 5. Increasing this to 10 resulted in all applications meeting the availability requirement.

B. Task Communication Energy

Figure 6 plots the communication energy of the previous 10 applications on the same MPSoC architecture. The energy numbers of all techniques are normalized with respect to minimum energy achieved for the given application on the given platform [14]. As can be seen from the figure, the communication energy of *SME* is least among all the existing fault-tolerant techniques. This is expected as communication energy is explicitly modeled in the optimization objective. The proposed technique minimizes this further by achieving 15% less communication energy, on average, than *SME* while staying within 5% of the minimum energy mapping of [14]. The energy savings with the proposed technique constitute 30% (18%) of the communication (total) energy of an application.

Although both *SME* and the proposed approach model communication energy explicitly in the objective, the savings in the proposed technique is more. This is because *SME* is based on single failure model; the minimum energy mapping can be potentially discarded if it violates the MTTF requirement with single-fault scenario, irrespective of the fact that the same mapping with task re-mapping can satisfy the constraint.

C. Task Migration Overhead

Figure 7 plots the migration energy of all the techniques for the same set of applications. Results are normalized with respect to *MT*. As can be seen from the figure, the migration

TABLE III
ALGORITHM SENSITIVITY TO DESIGN PARAMETER m

m	Total Mappings Evaluated	Applications Missing Availability Constraint	Percentage Violation
1	15,625	12	0.5%
5	101,556	4	0.05%
10	1,888,887	0	0%

overhead in the proposed approach is the least with an average 50% savings as compared to the existing research.

From the results of the previous three sections it can be concluded that proper task mapping can minimize communication energy and migration overhead while satisfying MPSoC availability requirement.

D. Complexity of the Proposed Algorithm

The complexity of Algorithm 1 is governed by line 2, 7 & 9. The number of mappings generated with $|V_{app}|$ tasks on n homogeneous cores (line 2) is $O\left(n \times |V_{app}|\right)$ (ref. [18]). In the worst case, the while loop (line 4-11) is traversed for each of these mappings. In each loop, the total mappings evaluated is equal to the number of nodes in the mapping tree. This is given by $\sum_{i=1}^{n-k+1} m^i$. Hence, the worst case complexity of Algorithm 1 is $O\left(n \times |V_{app}| \times m^{n-k+1}\right)$.

E. Distance from Optimality

Figure 8 plots the percentage deviation of the proposed approach from the energy minimum point at center (obtained by solving the problem at hand using standard convex solvers) for the same set of applications. As can be seen from the figure, the proposed technique is within 10% of the minimum energy. Thus, solving convex optimization, communication energy can be reduced further by 10% (over that obtained in Section VII-B). The price paid is 3 fold execution-time. This trade-off is omitted here for space limitations.

F. Sensitivity of the Proposed Algorithm

Table III reports the sensitivity of the proposed algorithm to the design parameter m. The table reports the number of mappings evaluated and the number of applications for which no mappings satisfied the availability requirement, for different values of m. The experiment is conducted with 45 synthetic and 5 real-life applications on an architecture with 9 cores (3×3). The number of tasks in the synthetic application is randomly generated with mean 20.

As can be seen from the table, for $m = 10$, the proposed algorithm generates a solution for all the applications considered. However, the number of mappings evaluated is large. This is due to the increased number of nodes of the mapping tree which results in a large number of paths from the root to the leaf nodes. Setting this value to 1 resulted in a substantial reduction of the number of mappings. However, 6 out of 50 applications missed the availability requirement. Finally, m =5 provides the best result with 4% misses (2 out of 50) and a reasonable number of mappings.

VIII. CONCLUSION

This paper introduces a technique to model the availability of an MPSoC based on Markov state transition. Based on this, a technique is proposed to minimize the communication and migration energy of the MPSoC guaranteeing to meet its availability requirement under the influence of reparable and intermittent device defects. Simulation results confirm that the proposed technique minimizes the communication energy by 30% and the migration overhead by 50%. Although, homogeneous cores are considered for experiments, an immediate extension of the approach is to consider heterogeneity of cores.

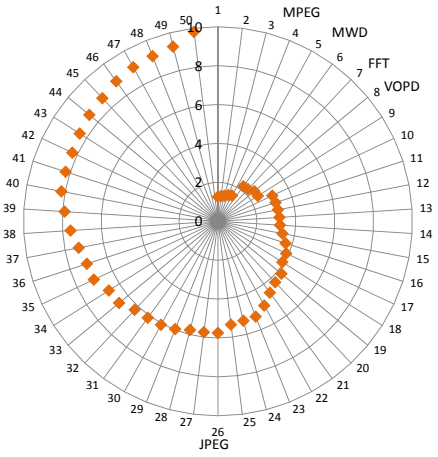

Fig. 8. Distance from optimality

ACKNOWLEDGMENT

This work was supported by Singapore Ministry of Education Academic Research Fund Tier 1 with grant number R-263-000-655-133.

REFERENCES

[1] C. Constantinescu, "Trends and challenges in VLSI circuit reliability," *IEEE Micro*, 2003.
[2] P. Wells *et al.*, "Adapting to Intermittent Faults in Future Multicore Systems," in *IEEE Conference on Parallel Architecture and Compilation Techniques (PACT)*, 2007.
[3] S. Pan *et al.*, "Ivf: Characterizing the vulnerability of microprocessor structures to intermittent faults," *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)*, 2012.
[4] E. B. Nightingale *et al.*, "Cycles, cells and platters: an empirical analysisof hardware failures on a million consumer PCs," in *ACM conference on Computer systems*, 2011.
[5] J. Huang *et al.*, "Analysis and optimization of fault-tolerant task scheduling on multiprocessor embedded systems," in *IEEE/ACM/IFIP Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, 2011.
[6] A. K. Coskun *et al.*, "Temperature aware task scheduling in MPSoCs," in *IEEE Conference on Design, Automation and Test in Europe (DATE)*, 2007.
[7] T. Chantem *et al.*, "Temperature-Aware Scheduling and Assignment for Hard Real-Time Applications on MPSoCs," in *IEEE Conference on Design, Automation and Test in Europe (DATE)*, 2008.
[8] L. Thiele *et al.*, "Thermal-aware system analysis and software synthesis for embedded multi-processors," in *ACM Design Automation Conference (DAC)*, 2011.
[9] L. Huang *et al.*, "Lifetime reliability-aware task allocation and scheduling for MPSoC platforms," in *IEEE Conference on Design, Automation and Test in Europe (DATE)*, 2009.
[10] S. Wang *et al.*, "Thermal-aware lifetime reliability in multicore systems," in *ISQED*, 2010.
[11] A. Hartman *et al.*, "A case for lifetime-aware task mapping in embedded chip multiprocessors," in *IEEE/ACM/IFIP Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, 2010.
[12] C.-L. Chou *et al.*, "FARM: Fault-aware resource management in NoC-based multiprocessor platforms," in *IEEE Conference on Design, Automation and Test in Europe (DATE)*, 2011.
[13] A. Das *et al.*, "Reliability-Driven Task Mapping for Lifetime Extension of Networks-on-Chip Based Multiprocessor Systems," in *IEEE Conference on Design, Automation and Test in Europe (DATE)*, 2013.
[14] J. Hu *et al.*, "Energy-aware communication and task scheduling for network-on-chip architectures under real-time constraints," in *IEEE Conference on Design, Automation and Test in Europe (DATE)*, 2004.
[15] B. Zhao *et al.*, "Generalized reliability-oriented energy management for real-time embedded applications," in *ACM Design Automation Conference (DAC)*, 2011.
[16] J. S. S. T. Association *et al.*, "Failure mechanisms and models for semiconductor devices," *JEDEC Publication JEP122-B*, 2003.
[17] A. Das *et al.*, "Fault-Aware Task Re-Mapping for Throughput Constrained Multimedia Applications on NoC-based MPSoC," in *IEEE Symposium on Rapid System Prototyping (RSP)*, 2012.
[18] A. Singh *et al.*, "A Hybrid Strategy for Mapping Multiple Throughput-constrained Applications on MPSoCs," in *ACM Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES)*, 2011.