Reliability-Aware Platform-Based Design Methodology for Energy-Efficient Multiprocessor Systems

Anup Das, Student Member, IEEE, Akash Kumar, Senior Member, IEEE, and Bharadwaj Veeravalli, Senior Member, IEEE,

Abstract—Lifetime reliability is an emerging concern in multiprocessor systems as escalating power density and hence temperature variation continues to accelerate wear-out leading to a growing prominence of device defects. In this paper, we propose a platform-based design methodology that involves performance-aware task mapping on a multiprocessor system to jointly minimize energy consumption and temperature related wear-out. Fundamental to this methodology is a simplified temperature model that incorporates not only the transient and steady-state behavior (temporal effect) but also its dependency on the temperature of the surrounding cores (spatial effect). The proposed temperature model is integrated in a gradient-based fast heuristic that controls the voltage and frequency of the cores to limit the average and peak temperature leading to a longer lifetime, simultaneously minimizing the energy consumption. Lifetime computation considers task remapping, which is a common feature available in modern multiprocessor systems. A linear programming approach is then proposed to distribute the cores of a multiprocessor system among concurrent applications to maximize the lifetime. Experiments conducted with a set of synthetic and real-life applications represented as synchronous data flow graphs demonstrate that the proposed approach minimizes energy consumption by an average 24% with 47% increase in lifetime. For concurrent applications, the proposed lifetime-aware core distribution results in an average 10% improvement in lifetime as compared to the performance-aware core distribution.

Index Terms—Lifetime reliability, mean time to failure (MTTF), platform-based design, synchronous data flow graphs.

I. INTRODUCTION

Lifetime reliability is a crucial design concern for modern multiprocessor systems as escalating power density and hence temperature variation continues to accelerate wear-out, leading to an increase in device defects. This has attracted significant attention both in the industry and academia to investigate platform-based design approaches, which involve system-level techniques such as mapping and scheduling of applications on a given multiprocessor platform to mitigate wear-out leading to an extended mean time to failure (MTTF) [1]–[5]. These studies do not minimize energy consumption, which is also crucial for battery-operated embedded devices.

Modern cores support a wide range of voltages and frequencies, which are often exploited to perform dynamic voltage and frequency scaling (DVFS) to minimize the task computation energy [6]–[9]. Furthermore, communication-aware task mapping heuristics minimize the task communication energy significantly. These have motivated researchers in recent years to jointly optimize lifetime reliability and energy consumption through intelligent task mappings [10]–[12]. The existing works on energy-reliability joint optimization suffer from the following two limitations – accuracy and scope.

Accuracy: Lifetime estimation and optimization at design-time require predicting the thermal behavior of an application. HotSpot [13] is the standard and the most prevalent thermal simulation tool used both in the industry as well as in academia. However, integrating this tool directly in the design space exploration process leads to a super exponential exploration time even for a moderate problem size (in terms of the number of tasks and cores). This is due to the large simulation time of the HotSpot tool. Some of the recent studies addressed this scalability problem by modeling the heat transfer phenomena using a resistive-capacitive (RC) equivalent model. The analytical solution to determine the temperature of a core from the RC equivalent model using mathematical techniques such as LU decomposition or sparse solver, are usually too slow to be incorporated in the design space exploration framework. Steady-state approximation simplifies the solution, but are accurate if the execution times of the tasks are comparable to the thermal time constant of the package, which is typically of the order of few hundreds of seconds. Finally, ignoring the spatial dependency leads to simplification of the RC equivalent model, but results in underestimation of the temperature and a corresponding overestimation of the mean time to failure.

Furthermore, most of the existing studies on lifetime reliability measure the mean time to failure as the time to the first fault. This is true for the following systems:

• a system without support for task-migration; and
• a system that always execute applications requiring all the cores of the architecture to satisfy their performance.

Modern multiprocessor systems often execute applications requiring fewer number of cores (for satisfying the performance constraint) than the total available cores. For these systems, tasks on a faulty core are migrated to other functional cores following fault detection. The existing techniques considering time to first failure leave a significant scope for reliability and energy improvement when applied to these systems.

Scope: The existing lifetime optimization techniques are all based on sequential execution of applications represented as directed acyclic graphs (DAGs) that are not sufficiently expressive to model streaming multimedia and other data flow applications. This class of applications requires support for modeling cyclic dependency, multi-input tasks, multi-rate tasks, pipelined execution and a natural way for dealing with latency and buffer requirements. As discussed in Section III, synchronous data flow graphs (SDFGs) allow more suitable modeling for these applications. The existing techniques for DAGs cannot be applied directly on SDFGs due to the cyclic actor dependencies and the overlapping of multiple iterations (pipelined) in the schedule. Furthermore, all the existing techniques determine lifetime optimum mapping for a single application. However, multiprocessor systems are often designed for multiple applications, many of them enabled concurrently.
(use-case\textsuperscript{1}). As shown in this paper, lifetime-aware distribution of the cores among the concurrent applications leads to a significant improvement in MTTF.

To address the limitations of the existing approaches, a temperature model is first proposed that is based on thermal characterization of a multiprocessor system using the HotSpot tool or any other thermal measurement approaches such as using sensors or performance counters. The model incorporates the following:

1. **temporal dependency** i.e., the relationship between the temperature of a core as a function of time and its dependency on the operating voltage and frequency; and

2. **spatial dependency** i.e., the influence of the neighboring core’s temperature on the temperature of a core.

A gradient-based fast heuristic is then proposed incorporating the temperature model to jointly optimize energy and lifetime reliability of a multiprocessor system with applications represented as SDFGs. The approach leverages on the native SDFT tool \cite{15}, and can be easily ported to applications represented as DAGs, making the approach generic for both streaming multimedia and non-multimedia applications. Following are the key contributions of this paper:

- a simplified temperature-model considering temporal and spatial dependencies;
- computing the MTTF considering task remapping;
- a gradient-based fast heuristic to jointly optimize lifetime reliability and energy;
- reliability optimization considering synchronous data flow graphs that support for modeling cyclic task dependency, multi-input tasks, multi-rate tasks and pipelined execution; and
- maximizing the MTTF of multiprocessor system using concurrent applications.

Some of these contributions appear in our earlier work \cite{12}. This work extends our earlier work by providing a detailed evaluation of the proposed temperature model, computing lifetime considering task remapping and reliability-aware core distribution for concurrent applications.

The remainder of this paper is organized as follows. Section II provides a brief overview of the state-of-the-art approaches on temperature prediction for reliability optimization. This is followed by the problem formulation in Section III and the proposed temperature model in Section IV. The temperature computation from a given SDFG schedule is demonstrated in Section V. The design methodology is discussed next in Sections VI. Experimental results are presented in Section VII and Section VIII presents the conclusions.

II. RELATED WORKS

System-level design techniques such as task mapping and scheduling have received significant research attention, focusing both on the unrestricted optimization (best effort) as well as the reliability- and energy-driven restricted counterpart. The reliability- and energy-aware task mapping techniques can be classified into two categories – on-line and off-line. The on-line techniques adapt the power control levers or the scheduling decisions based on the continuous feedback from the hardware performance monitoring unit at run-time. The objective is to minimize energy \cite{6} or maximize reliability \cite{16}.

The off-line techniques analyze applications at design-time to determine energy and reliability optimum task mappings. Since this work falls into off-line category, some of the off-line design techniques are reviewed in details.

A task mapping technique is presented in \cite{7} to minimize the computation and the communication energy of multimedia applications on multiprocessor systems. A co-design technique is presented in \cite{8} to determine the voltage and frequency of cores considering the mode execution probabilities. The technique in \cite{9} models the dynamic behavior of an application as a scenario-aware dataflow graph. Based on this, a technique is proposed to determine the voltage and frequency of the cores of a multiprocessor system. These techniques do not incorporate the influence of voltage and frequency reduction on lifetime reliability.

Since temperature has a significant impact on device wear-out, there are quite some research studies on offline task allocations techniques for temperature minimization. A thermal-aware task mapping and scheduling technique is proposed in \cite{17}. Steady-state temperature is generated using the HotSpot \cite{13} tool from the average power of the task schedule. A mixed integer linear programming (MILP)-based task mapping and scheduling for peak temperature minimization is proposed in \cite{18}. A convex optimization formulation of the voltage-frequency dependency of temperature is formulated in \cite{19}. An interior-point algorithm is proposed to solve the convex problem. A fast event-driven approach is proposed in \cite{20} to estimate the temperature of multiprocessor systems. The estimation overhead is minimized using prebuilt lookup tables and predefined leakage calibration parameters. All these techniques determine the steady-state temperature only. The temperature from these approaches are accurate if the execution time of a given application is of the order of the thermal time constant of the package, which is typically hundreds of seconds. Very few thermal management techniques exist in literature that account for both the transient and the steady-state phases. A temperature-aware technique is proposed in \cite{21} to distribute the idle time in order to control the power consumption and hence the temperature. The proposed temperature model incorporates the dynamic and the leakage power components and therefore results in fairly accurate temperature estimation. A similar approach is proposed in \cite{22}. However, both these techniques consider a uni processor system and results in scalability and spatial dependency issues when applied to multiprocessor systems.

Different wear-out mechanisms are influenced by temperature differently. Hence, there are studies that optimize lifetime reliability directly considering these wear-out mechanisms through intelligent task mapping and scheduling. A reliability estimation technique is proposed in \cite{1} for application specific multiprocessor systems to consider multiple failures by incorporating the effect of faults on the subsequent fault rates. A simulated annealing based technique is proposed in \cite{2} to maximize the lifetime reliability of a multiprocessor system. The steady-state temperature values is determined using the HotSpot tool for all combinations of the active tasks on different processors. These temperature data are stored in a lookup and used during the optimization step. Ant-colony based optimization technique is proposed in \cite{3} to determine the task mapping that maximizes lifetime defined as the time to the first failure. This technique has shown that the lifetime of a multiprocessor system using temperature-aware optimization technique can be significantly lower than when lifetime is
explicitly optimized. However, not enough details are provided on the thermal model used to estimate the temperature to determine the wear-out related aging. A convex optimization based approach is proposed in [4] to maximize the lifetime reliability of the cores of a multiprocessor system considering electromigration related wear-out mechanism. The proposed approach also incorporates the wear-out of the underlying networks-on-chip using a simplistic model to determine the steady-state temperature. A resource management technique is proposed in [5] to minimize processor wear-out, simultaneously providing tolerance for transient and intermittent faults. The underlying temperature model is also similar to [2]. A simulated annealing based energy-reliability joint optimization technique is proposed in [11] based on the temperature model of [2]. A common limitation of all these reliability-aware optimization techniques is that the transient phase of the temperature is ignored resulting in an inaccuracy in lifetime estimation. To address this issue, the technique in [11] uses eigen-value decomposition based approach to determine the steady-state dynamic temperature profile. The proposed approach is shown to be the most accurate among all the existing techniques to determine not only the peak or average temperatures but also the thermal amplitude and frequency, which is crucial for the optimization of thermal cycling related lifetime. However, this technique requires detailed specification of RC circuits, which is sometimes difficult to obtain for the commercial off-the-self (COTS) multiprocessor systems. Table I summarizes the related works on reliability optimization.

### III. PROBLEM FORMULATION

#### A. Application Model

Synchronous Data Flow Graphs (SDFGs, see [27]) are often used for modeling modern DSP applications and for designing concurrent multimedia applications implemented on a multi-processor system-on-chip. Both pipelined streaming and cyclic dependencies between tasks can be easily modeled in SDFGs. SDFGs allow analysis of a system in terms of throughput and other performance properties, e.g. latency, buffer requirements [28].

The nodes of a SDFG are called actors; they represent functions that are computed by reading tokens (data items) from their input ports and writing the results of the computation as tokens on the output ports. The edges in the graph, called channels, represent dependencies among different actors.
• **actor distribution**: determine the assignment of the actors of the SDFG onto the cores of the multiprocessor system;

• **operating point**: determine the voltage and frequency of the cores for executing the actors of the SDFG.

For the ease of problem representation, two variables $x_{i,j}$ (representing the **actor distribution**) and $y_{i,k}$ (representing the **operating point**) are defined as follows:

$$x_{i,j} = \begin{cases} 1 & \text{if actor } a_i \text{ is executed on core } c_j \\ 0 & \text{otherwise} \end{cases}$$

$$y_{i,k} = \begin{cases} 1 & \text{if actor } a_i \text{ is executed at operating point } (V_k, \omega_k) \\ 0 & \text{otherwise} \end{cases}$$

Constraints on these variables are set such that an actor is mapped to only one core at a single operating point. Thus,

$$\sum_{j=0}^{N_c-1} x_{i,j} = 1 \quad \text{and} \quad \sum_{k=0}^{N_f-1} y_{i,k} = 1 \quad \forall a_i \in \mathcal{A}$$

The **actor distribution** and **operating point** of SDFG are represented as two matrices:

$$M_d = \begin{pmatrix} x_{0,0} & x_{0,1} & \cdots & x_{0,N_c-1} \\ x_{1,0} & x_{1,1} & \cdots & x_{1,N_c-1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N_c-1,0} & x_{N_c-1,1} & \cdots & x_{N_c-1,N_c-1} \end{pmatrix}$$

(2)

$$M_o = \begin{pmatrix} y_{0,0} & y_{0,1} & \cdots & y_{0,N_f-1} \\ y_{1,0} & y_{1,1} & \cdots & y_{1,N_f-1} \\ \vdots & \vdots & \ddots & \vdots \\ y_{N_f-1,0} & y_{N_f-1,1} & \cdots & y_{N_f-1,N_f-1} \end{pmatrix}$$

(3)

The core to which actor $a_i$ is assigned is denoted by $\phi_i$ and is given by $\phi_i = X_i \times N_c$, where $X_i = (x_{i,0}, x_{i,1}, \cdots, x_{i,N_c-1})$ and $N_c$ is the matrix of integers from 0 to $N_c$ i.e. $N_c = (0, 1, \cdots, N_c-1)^T$. The operating point of actor $a_i$ is denoted by $\theta_i$ and is given by $\theta_i = Y_i \times N_f$, where $Y_i = (y_{i,0}, y_{i,1}, \cdots, y_{i,N_f-1})$.

### D. Wear-out Modeling

The lifetime reliability of a core at time $t$ is defined as the probability of correct operation of the core up to time $t$ and is given by $R(t) = e^{-(t \cdot r)^\beta}$, where $r$ is the aging of the core per iteration and is given by (refer [2], [30]).

$$r = \frac{1}{t_p} \sum_{i \in \mathcal{P}} \Delta T_i$$

(4)

where $t_p$ is the period of the application graph, $\alpha(T^o)$ is the fault density (typically Weibull or Lognormal distribution) and $T^o$ is the average temperature in the interval $\Delta T_i$. This equation allows to model any wear-out effects such as electro-migration and negative bias temperature instability [2], [12].

The MTTF of core $c_j$ with reliability $R_j(t)$ is given by the area under the reliability curve as

$$\text{MTTF}_j = \int_0^\infty R_j(t)dt = \int_0^\infty e^{-(t \cdot r)^\beta} dt$$

(5)

For a multiprocessor system with multiple interconnected cores and support for task migration, the MTTF of the system is defined as the time to $F$ faults with the tasks on the faulty cores being migrated to other functional cores after every failure. Here

$$F = \begin{cases} N_c & \text{for a graceful degrading system} \\ N_c - N^{\text{min}} + 1 & \text{otherwise} \end{cases}$$

(6)

![Fig. 3. MTTF computation with different temperature profile.](image)

where $N^{\text{min}}$ is the minimum number of cores required to satisfy the performance constraint for an application. In this work, we consider a system to have failed if an application’s performance constraint cannot be satisfied with the available functional cores. Hence, $N^{\text{min}} \geq 1$.

To demonstrate the MTTF computation considering task remapping, an example is provided with three cores (core 0, 1 and 2). For simplicity, let $N^{\text{min}} = 1$. The initial schedule $S_0$ uses all the three cores and stresses core 2 more than the other two cores. The reliability curves for the three cores are shown in Figure 3. Core 2 has the least lifetime and it breaks at time $\tau_0$. As established in Section II, the previous works on lifetime reliability defines MTTF to be the time to the first failure, hence the MTTF for these works is $\tau_0$. At time $t = \tau_0$, a second schedule $S_1$ (using core 0 and 1) is applied. The change in the reliability profile of core 0 and core 1 are due to different wear-out, which can be attributed to the difference in temperature from this new schedule. The new schedule stresses core 0 more than core 1 and therefore core 0 breaks at time $t = \tau_1$. At this time, all the actors are remapped to core 1 and are ordered honoring the actor dependency. This schedule is identified in the figure as $S_2$ and results in reliability profile of $r_{1}$. With this new reliability profile, core 1 breaks at time $t = \tau_2$. The lifetime (MTTF) of the system is therefore $\tau_2$.

Besides MTTF, another interesting metric for multiprocessor system supporting task remapping is **processor years**, defined as the aggregate utilization of the different cores of the system over the entire lifetime. For the above example, this is calculated as follows. For the interval 0 to $\tau_0$, all three cores are active; for the interval $\tau_0$ to $\tau_1$ two core are active; and for the interval $\tau_1$ to $\tau_2$ only one core is active. Assuming the time in this figure are all in years, the **processor years** of the above system is $3 \cdot \tau_0 + 2 \cdot (\tau_1 - \tau_0) + 1 \cdot (\tau_2 - \tau_1)$.

### E. Energy Modeling

Actor-level voltage and frequency scaling is assumed for this work i.e., every actor of a SDFG is associated with a voltage and frequency values that are set on the core executing the actor. The energy consumed on a multiprocessor system while executing the SDFG consists of the following components:

• **computation energy**: dynamic and leakage energy consumed on the cores due to the actors execution; and

• **communication energy**: dynamic and leakage energy consumed on the network-on-chip (NoC) due to the data communication among the connected actors.
A point to note here is that the leakage energy consumed on the NoC is dependent on the NoC type and the topology. For this work, a spatial division multiplexing-based NoC is assumed and therefore, the leakage power consumed on the NoC is negligible [31].

**Dynamic Energy of SDFG:** The dynamic energy of a SDFG is given by the following formula [32].

\[ E_{\text{dyn}} = E_{\text{tr}} + N_{\text{iter}} \cdot E_{\text{ss}} \]

where \( E_{\text{tr}} \) is the actor dynamic energy in the transient phase of the schedule, \( E_{\text{ss}} \) is the actor dynamic energy per iteration of the steady state phase and \( N_{\text{iter}} \) is the number of iterations of the steady state phase. Usually, the number of steady state iterations (i.e. \( N_{\text{iter}} \)) is a large number (e.g., periodic decoding of video frames) and hence for all practical purposes, the dynamic energy of the steady state phase dominates over that in the transient phase. The rest of this paper, computation (or communication) energy implies computation (or communication) energy of the steady state phase per iteration.

The dynamic energy consumed by an actor \( a_i \) at the operating point \( k \) is given by

\[ e_{\text{dyn}}(i,j,k) = C_{eff} \cdot \beta \cdot V^2 \cdot Q_k \cdot t_{ij} \cdot Rpt[a_i] \]  \hspace{1cm} (7)

where \( \beta \) is the activity factor, \( C_{eff} \) is the effective load capacitance, \( t_{ij} \) is the execution time of actor \( a_i \) on core \( c_j \) at operating point \( k \) (i.e. operating voltage \( V_k \) and operating frequency \( \omega_k \)) and \( Rpt[a_i] \) is the number of firings of actor \( a_i \) per steady state iteration of the SDFG. The total dynamic energy per steady-state iteration of the SDFG is

\[ E_{\text{dyn}} = \sum_{a_i \in A} e_{\text{dyn}}(i,\phi_i,\theta_i) \]  \hspace{1cm} (8)

**Leakage Energy of SDFG:** The leakage energy of core \( c_j \) consumed during the execution of actor \( a_i \) at operating point \( k \) is given by the following formula [32].

\[ e_{\text{leak}}(i,j,k) = N_{\text{gates}} \cdot V_k \cdot I_0 \cdot \left[ AT^2 \cdot e^{\gamma V_k + \beta} + Be^{-V_k + \delta} \right] \cdot t_{ij} \cdot Rpt[a_i] \]  \hspace{1cm} (9)

where \( N_{\text{gates}} \) is the number of gates of the core, \( I_0 \) is the average leakage current and \( A, B, \alpha, \beta, \gamma, \delta \) are technology dependent constants (refer to [32]) and \( T \) is the average temperature of the actor during the steady-state iteration. The total leakage energy is

\[ E_{\text{leak}} = \sum_{a_i \in A} e_{\text{leak}}(i,\phi_i,\theta_i) \]  \hspace{1cm} (10)

**Dynamic Energy on the NoC:** In [33], bit energy (\( E_{\text{bit}} \)) is defined as the energy consumed in transmitting one bit of data through the routers and links of a NoC.

\[ E_{\text{bit}} = E_{\text{bit,trans}} + E_{\text{bit,link}} \]  \hspace{1cm} (11)

where \( E_{\text{bit,trans}} \) and \( E_{\text{bit,link}} \) are the energy consumed in the switch and the link, respectively. The energy per bit consumed in transferring data between cores \( c_p \) and \( c_q \), situated \( \eta_{\text{hop}}(p, q) \) away is given by Equation 12 according to [7].

\[ E_{\text{bit}}(p, q) = \begin{cases} \eta_{\text{hop}}(p, q) + 1 \cdot E_{\text{bit,trans}} + \eta_{\text{hop}}(p, q) \cdot E_{\text{bit,link}} & \text{if } p \neq q \\ \text{otherwise} \end{cases} \]  \hspace{1cm} (12)

The dynamic energy consumed on the NoC is therefore given by Equation 13 where \( \phi_i \) and \( \phi_i' \) are the cores where actors \( a_i \) and \( a_i' \) are mapped, respectively.

\[ E_{\text{dia}} = \sum_{a_i \in A} d_{ij} \cdot E_{\text{bit}}(\phi_i, \phi_i') \]  \hspace{1cm} (13)

The total energy is

\[ E_{\text{tot}} = E_{\text{core}} + E_{\text{leak}} + E_{\text{dia}} \]  \hspace{1cm} (14)

**F. Reliability-Energy Joint Metric**

To jointly optimize lifetime reliability and energy, a single metric lifetime quotient is defined as

\[ \frac{MTTF}{E^{tot}} \]  \hspace{1cm} (15)

The optimization objective is to maximize \( l_q \).

**IV. PROPOSED TEMPERATURE MODEL**

**A. Background on Temperature Modeling**

Temperature of a multiprocessor system is usually determined using a RC equivalent thermal model. The temperature of a single core is related to its power dissipation according to the following equation [13].

\[ C \frac{dT(t)}{dt} + G(T(t) - T_{amb}) = P(t) \]  \hspace{1cm} (16)

where \( C \) is the thermal capacitance, \( G \) is the thermal conductance, \( t \) is the time, \( T_{amb} \) is the ambient temperature, \( T(t) \) is the instantaneous temperature and \( P(t) \) is the instantaneous power that is composed of the dynamic and the static components. The dynamic power is dependent on the voltage and frequency of operation and the static power is dependent on the temperature (refer to Section III-E). The solution to the above equation consists of transient and steady-state phases. In the transient phase, the temperature increases with time up to a point beyond which, the steady-state phase settles in and the temperature saturates to its steady-state value. The core’s wear-out is dependent on its operating temperature, which needs to incorporate both the transient and the steady-state phases.

Referring back to Figure 2, the temperature of any core, say core \( c_i \), of a system with interconnected cores depends on:

- A.1 The time of execution of an actor on \( c_i \).
- A.2 The voltage and frequency of \( c_i \).
- A.3 The temperature of the cores surrounding \( c_i \).

Thus, A.1 and A.2 represents the temporal dependency and A.3 represents the spatial dependency. For such a system, the temperature, power, thermal capacitance and thermal conductance in Equation 16 are all vectors. The transient and steady-state values can be obtained by solving the above equation analytically. The solution to the differential equation is

\[ T(t) = e^{\kappa t}T(0) + \kappa^{-1}(e^{\kappa t} - I) C^{-1} \mathbf{P}(t) \]  \hspace{1cm} (17)

where \( \kappa = -C^{-1}G \), \( T(0) \) is the initial temperature and \( I \) is the identity matrix. The direct solution technique using LU decomposition or sparse solver is usually slow and results in an exponential design space exploration time. Although the iterative solution in [11] simplifies the solution, the execution time is still exponential when applied for temperature prediction using multi-application use-cases.

A simplification to the analytical approach is to ignore the spatial dependency by considering the temperature for cores individually, such as the one proposed in [21]. To signify the importance of temperature underestimation by ignoring the spatial dependency (component A.3), an experiment is conducted using the HotSpot tool to measure the steady-state temperature. The multiprocessor architecture used for the HotSpot tool is shown in Figure 2 with the specifications of the cores reported in Table II. The temperature is determined by setting the power dissipation of core \( c_1 \) as 0, with a constant power corresponding to the operating point OPP1G (i.e. 1.35V, 1GHz) set for the one-hop and the two-hop neighboring cores (cores \( c_1 \) through \( c_4 \) are the one-hop neighbors and cores \( c_5 \) through \( c_7 \) are...
Temperature considering A.1 and A.2

Temperatures of core i

Temperature model is based on: analysis technique is proposed in this paper. The proposed

There are two bars shown on the plot. The left bar for each label corresponds to the temperature of core \( i \) with all the cores as idle. The right bar corresponds to the temperature of core \( i \) with the neighboring core's activity. The label \( \text{V} \) obtained

The solution to the differential equation is represented as

\[
T_i(t) = f(V_i, \omega_i, t) + g(\{T_j(t) | \forall c_j \in \mathbb{N}(c_i)\})
\]

where \((V_i, \omega_i)\) are the voltage and frequency of core \( c_i \), \( t \) is the time and \( \mathbb{N}(c_i) \) are the cores in the neighborhood of \( c_i \). The function \( f \) and \( g \) represent the temporal and spatial dependency, respectively and are derived in two steps.

1) Determination of \( f \): The function \( f \) can be determined using two alternative approaches – by solving Equation 16 directly for a processing core; or by simulation using the HotSpot tool for the power consumption corresponding to different operating points of the processing core to capture the transient and steady-state behaviors, as shown in Figure 5.

2) Characterization for \( g \): The temperature data for characterizing the function \( g \) are obtained as follows. The core \( c_i \) is set to idle mode and the operating points for the neighboring cores are varied. Performing exhaustive temperature simulations for different voltage-frequency combinations of all neighbors (one-hop neighbors, two-hop neighbors etc.) is time consuming, but only required once during the characterization step. A first order of approximation involves considering the voltage-frequency of only the immediate neighbors i.e. the east, west, north and south neighbors of a core referred to as \((V_i, \omega_e), (V_i, \omega_w), (V_i, \omega_n)\) and \((V_i, \omega_s)\), respectively with all other neighbors set to operate at the highest operating point. This is shown in Figure 6 where the voltage point is only shown for clarity of representation. The figure plots the temperature of core \( c_i \) as its voltage \( V_i \) is increased from 0.93V to 1.35V for few of these neighboring voltage combinations.

The temperature data are fed to the Matlab regression toolbox to derive the temperature model.

\[
T_i(t) = f(V_i, \omega_i, t) + g(\{V_j, \omega_j | \forall c_j \in \mathbb{N}(c_i)\})
\]

The proposed temperature model is determined once during the characterization process. The final expression for temperature (Equation 19) can be easily integrated in the design space exploration framework. However, the proposed model incorporates pessimism in three forms – separating the temporal and spatial dependency; characterizing the spatial dependency with the steady-state temperature of the nearest neighbors; and characterizing the spatial dependency with the

---

**B. Proposed Regression Model**

To provide a simplified solution of Equation 18, a regression analysis technique is proposed in this paper. The proposed temperature model is based on:

- temperature characterization data to incorporate the temporal dependency (capturing both the transient and steady-state behaviors); and
- temperature characterization data to incorporate the spatial temperature dependency.

![Fig. 4. Temperature underestimation ignoring the spatial dependency.](image)

![Fig. 5. Characterization for temporal dependency.](image)

![Fig. 6. Characterization for spatial dependency.](image)
non-nearest neighbors set to operate at the highest voltage and frequency. These pessimism lead to a temperature overestimation by as much as up to 6°C for individual applications (refer to Section VII). However, as established in Section VI-B, this temperature overestimation simplifies the reliability optimization for multiple simultaneous applications, which is a common requirement for multiprocessor systems.

C. Adaptation for COTS Multiprocessor Systems

The proposed regression technique is developed for the multiprocessor system using the HotSpot tool. However, the temperature characterization process is generic and can be applied to any commercial off-the-self multiprocessor systems with temperature measurement using thermal sensors or temperature prediction using performance monitoring unit.

V. TEMPERATURE COMPUTATION FROM A SCHEDULE

Figure 7 shows an example of a SDFG with four actors allocated on a platform with three cores. The schedule corresponding to a particular allocation is also shown in the same figure. We demonstrate the temperature computation for core 0 using this schedule. The temperature for other cores can be determined in a similar fashion. The time duration 0 − t_0 is divided into seven intervals by putting a time stamp at the instances when an actor starts or ends firing.

Core 0: Interval (0 → t_0)
In this interval, core 0 executes actor A at operating point (V_A, ω_A). The temperature at time t considering temporal effect is S(T_{\text{temp}} A, ω_A, t). The temperature considering the spatial effect is due to the idle voltages of core 1 and 2 and is given by S(V_{idle 1}, V_{idle 2}). The average temperature in this interval is

\[ T_0(0, t_0) = \frac{\int_0^{t_0} f(V_A, \omega_A, t)dt + g(V_{idle 1}, V_{idle 2})}{t_0 - 0} \]  

Core 0: Interval (t_0 → t_1)
In this interval, core 0 executes the first instance of actor B. Note in the SDFG, when actor A fires, it produces 3 tokens on the channel from actor A to actor B and one of these tokens is consumed for each firing of actor B. Therefore, there are three firing of actor B (indicated in the figure by B1, B2 and B3). The temperature at time t due to the temporal effect of actor B is f(V_B, \omega_B, t) and the temperature due to spatial effect is g(V_D, V_C). The average temperature is

\[ T_0(t_0, t_1) = \frac{\int_{t_0}^{t_1} f(V_B, \omega_B, t)dt + g(V_D, V_C)}{t_1 - t_0} \]  

Core 0: Interval (t_1 → t_2)
The temperature computation in this interval is similar to that in the interval (t_0 → t_1) and is given by

\[ T_0(t_1, t_2) = \frac{\int_{t_1}^{t_2} f(V_B, \omega_B, t)dt + g(V_D, V_C)}{t_2 - t_1} \]  

Core 0: Interval (t_2 → t_3)
During the execution of actor B3, there is a change in temperature profile due to the completion of actor D1 and the interval before actor D2 is executed. Hence, the execution time of actor B3 is split into two intervals (t_2 → t_3) and (t_3 → t_4).

The temperature computation in the interval (t_2 → t_3) is similar to that in the interval (t_0 → t_1)

\[ T_0(t_2, t_3) = \frac{1}{t_3 - t_2} \int_{t_2}^{t_3} f(V_B, \omega_B, t)dt + g(V_D, V_C) \]  

Core 0: Interval (t_3 → t_4)
The average temperature in this interval is given by

\[ T_0(t_3, t_4) = \frac{1}{t_4 - t_3} \int_{t_3}^{t_4} f(V_B, \omega_B, t)dt + g(V_{idle}, V_C) \]  

Core 0: Interval (t_4 → t_5)
In this interval, the temporal effect is due to the idle temperature of the core and is denoted by T_0^{idle}. The average temperature is given by

\[ T_0(t_4, t_5) = T_0^{idle} + g(V_D, V_C) \]  

Core 0: Interval (t_5 → t_6)
The temperature in this interval is given by

\[ T_0(t_5, t_6) = T_0^{idle} + g(V_{idle}, V_C) \]  

Reliability of Core 0
Combining these equations, the aging of core 0 is

\[ r_0 = \frac{1}{t_6} \sum_{i=0}^{t_6} \frac{t_i - t_{i-1}}{\alpha(T_0(t_i, t_{i-1}))} \]  

VI. DESIGN METHODOLOGY

The design methodology consists of two phases – analysis at design-time (consisting of the application and use-case optimizations) and execution at run-time. The design-time methodology is highlighted in Figure 8. The run-time manager is not part of our contribution, but is shown here for completeness. There are two databases for the multiprocessor system – the set of applications (S_{app}) and the set of use-cases (S_{use}). The proposed approach is to determine the actor distribution and the operating point (refer to Section III) for every application using n = N_{min}^m to N_c cores of the system. Thus, S_{app} \cdot (N_c - N_{min}^m + 1) optimization problems are solved at design-time. This is performed in the REOpt block. The solution consists of the actor distribution and operating
The minimum time corresponds to the failure of the most stressed core. The reliability profiles are shifted to account for the throughput constraint. The number of cores are stored in the mapping database. This application is executed on the throughput satisfaction and mapping database.

Algorithm 1: Generate Reliability and Energy Aware Mappings

Input: Application set $S_{app}$ and multiprocessor system $G_{arc}$
Output: MapDB and ThiRiCoDB
1: for all Application $A_i \in S_{app}$ do
2: $[G_{app}, T_a] = GetSDFG(A_i)$ //Get the corresponding SDFG and the throughput constraint.
3: Determine $N_{C}^{min}$, the minimum number of cores required for satisfying the throughput requirement.
4: for $n = N_{C}^{min}$ to $N_c$ do
5: $(M_d, M_o) = REOpt(G_{app}, G_{arc}, T_a)$ //Perform individual application optimization
6: $[S, T] = MSDF^3(M_d, M_o, G_{app}, G_{arc})$ //Calculate the schedule and the throughput of the SDFG.
7: MapDB(i, n) = $(M_d, M_o, S)$ //Store the matrix in the mapping database.
8: $M = CalculateMTTF(i, n, N_{C}^{min}, MapDB)$
9: ThiRiCoDB(i, n) = $(M, M) //Store the throughput and MTTF values for the use-case optimization step.
10: end for
11: end for

Algorithm 2: Calculate the mean time to failure

Input: Application id $i$, the core index $n$, the minimum number of cores for throughput satisfaction and mapping database $MapDB$
Output: $MTF$ matrix
1: Initialize $tff = 0$ and $ri = n$
2: while $ri \geq N_{C}^{min}$ do
3: $(M_d, M_o, S) = MapDB(i, ri)$ //Fetch the values.
4: Determine reliability profiles from $S$ as demonstrated in Section V
5: Shift the reliability profiles by $tff$
6: Determine $t$, the time to failure of the most stressed core
7: $tff = tff + 1$ and $ri = ri - 1$
8: end while

point matrices stored in the MapDB database and the three-dimensional (3D) vector – throughput, reliability (MTTF) and core count stored in the ThiRiCoDB database.

Algorithm 1 provides the pseudo-code of the design flow. For every application $A_i$ of the set $S_{app}$, the corresponding SDFG representation and the throughput constraint are fetched from the database. This application is executed on the multiprocessor system with $n$ cores identified as $G_{arc}$, where $n$ is varied from $N_{C}^{min}$ to $N_c$. The reliability–energy joint optimization is first performed on the application (line 5) to obtain the actor distribution matrix $M_d$ and the operating point matrix $M_o$. These are stored in the MapDB (line 7). The actor distribution is used in the $MSDF^3$ tool that leverage on the native $SDF^3$ tool [15] to generate the throughput and schedule. The schedule thus obtained is used in the $CalculateMTTF()$ routine (line 8) to compute the MTTF. The throughput and the MTTF values corresponding to the number of cores are stored in the ThiRiCoDB for the use-case optimization step that addresses core distribution among concurrent applications.

The $CalculateMTTF$ routine determines the MTTF in an iterative manner as shown as pseudo-code in Algorithm 2. A running index $ri$ is maintained to index the schedule with one less core. At the start of the iteration, the mapping and the scheduling are fetched from the $MapDB$. The schedule is used to compute the reliability profile of every cores of the system. The reliability profiles are shifted to account for the aging already encountered in the cores. The times to failure for all the cores are determined using Equation 5. The minimum time corresponds to the failure of the most stressed core. This is added to the $tff$ and the running index is decremented.

A. Reliability Optimization for Individual Application

The objective function (lifetime quotient) of the optimization problem is non-linear; a gradient-based fast heuristic is proposed to solve it. This is shown as pseudo-code in Algorithm 3. The algorithm starts from an initial allocation computed using the native $SDF^3$ tool (line 2). Subsequently, the algorithm remaps every actor to every core to determine a priority that increases the lifetime of the system.

The $REOpt()$: Reliability and Energy Optimization for an Application

Input: $S_{app}$, $G_{arc}$ and throughput constraint $T_a$
Output: Actor distribution and operating point matrices ($M_d$, $M_o$), which maximize $lq$
1: Initialize $M_o = (0 \ \ 0 \ \ \ldots \ \ 1)$ //Initialize the actors to the highest operating point.
2: $[M_d, S, T] = SDF^3(S_{app}, G_{arc})$ //Mapping, schedule and throughput using the native $SDF^3$ tool.
3: while true do
4: $P_{temp} = 0, P_{best} = M_{d}, best _{found} = false$ //Initialize the best values.
5: $lq = CalculateLQ(M_d, M_o, S, T)$ //Calculate the initial lifetime quotient.
6: for all $a_i \in A$ do
7: for all $c_i \in C$ do
8: for all $k \in [0, N_t - 1]$ do
9: $M_{temp} = M_{d}, M_{temp} = M_{o}$ //A temporary allocation matrix is used.
10: Update $M_{temp}, M_{temp}$ using $x_{i, j} = y_{i, j} = 1$ and $x_{i, l}$ = $y_{i, m} = 0, \forall i \neq j$ and $\forall m \neq k$
11: $[S_{temp}, T_{temp}] = SDF^3(M_{temp}, T_{temp}, S_{app}, G_{arc})$ //New schedule is computed using modified $SDF^3$.
12: $lq_{temp} = CalculateLQ(M_{d}, M_{o}, S_{temp}, T_{temp})$ //Calculate the new lifetime quotient.
13: Compute $P$ using Equation 28
14: if $T_{n} > T_{a}$ and $P > P_{best}$ then
15: $P_{best} = P, P_{temp} = M_{temp}, M_{o} = M_{o}$, $best_{found} = true, T = T_{n}$
16: end if
17: end for
18: end for
19: if $best_{found}$ then
20: $M_{d} = M_{temp}, M_{o} = M_{best}$ //Actor distribution and operating point matrices are updated with the best values found thus far.
21: else
22: break
23: end if
24: end while
25: Return ($M_d, M_o$) //Actor distribution and operating point matrices are returned.

Two conditions are considered in the priority computation: if the throughput of the current allocation ($T_{n}$) is lower than the original throughput ($T_a$), a gradient function is used to calculate its priority i.e. assignments that increase the lifetime quotient with the least throughput degradation are given higher priorities. Conversely, if the current throughput is higher than the original one, higher priorities are given to assignments with the largest increase in the lifetime quotient.

The algorithm remaps actor $a_i$ to a core $c_j$ at a frequency $\omega_k$ (lines 6 - 8). The actor distribution and the operating point of actor $a_i$ are changed (line 10). These matrices are used by the $MSDF^3$ tool to compute the throughput and schedule corresponding to the allocation $M_{temp}$. The $CalculateLQ$ function computes the lifetime quotient using Equation 14 to compute the energy and Algorithm 2 to compute the MTTF. The algorithm computes the priority function (line 13). If this priority is greater than the best priority obtained thus far and the throughput constraint is
satisfied, the best values are updated (line 15). The algorithm continues to remap as long as an assignment can be found without violating the throughput requirement. When no such remapping is possible, the algorithm terminates.

B. Reliability Optimization for Use-cases

In this section, the use-case level optimization problem is formulated based on the results obtained in Section VI-A. It is to be noted that when multiple applications are enabled simultaneously, the temperature due to the execution of one application is dependent not only on the temperature of the cores on which it is executed but also on the temperature due to other applications executing simultaneously. As a result, the wear-out (or the MTTF) due to single application can be significantly different than the actual wear-out (or the MTTF) for use-cases. This limitation is addressed using the pessimism introduced in the temperature model. Specifically, to determine the temperature for different cores during single application mode, all unused cores in the architecture (those, which can potentially execute other applications in multi-application use-case scenario) are considered to be operating and thus their temperature effect are incorporated in determining the temperature of the actual operating cores. Although this gives a pessimistic bound on the temperature and hence the reliability, the approach simplifies the problem solution for multi-application use-cases.

As established previously, the ThRiCoDB contains 3D databases with throughput and MTTF number for every core count of every application. The problem addressed here is to merge these 3D databases for applications enabled simultaneously such that the distribution of cores among these applications maximizes the system MTTF. For the ease of problem formulation, the following notations are defined:

\[ A_1, \ldots, A_n \]
\[ z_i \]
\[ M_i \]
\[ T_i \]

1) Formulation: The optimization problem is

\[
\text{maximize } \min_i \{M_i\}
\]

subject to

\[
\sum_{i=1}^{n} z_i \leq N_c
\]

\[
\forall i, T_i \geq \text{throughput constraint of } A_i
\]

2) Solution: Algorithm 4 provides the pseudo-code to solve Equation 29. A list is defined (\( RiList \)) to store the applications (their ID) of the use-case, the number of cores dedicated to it and the corresponding MTTF value. For every core in the system (line 3), the \( RiList \) is sorted to determine the application with the least MTTF (lines 4 - 5). A core is dedicated to this application (line 6) and the corresponding MTTF is fetched from the ThRiCoDB (line 7) and the \( RiList \) is updated.

VII. EXPERIMENTS AND DISCUSSIONS

Experiments are conducted with real-life and synthetic SDFGs on a multiprocessor system with cores arranged in a mesh architecture. The synthetic SDFGs are generated using the SDF3 [15] tool with the number of actors ranging from nine to twenty-five. These encompass both computation and communication dominated applications. The real-life SDFGs are H.263 Encoder, H.264 Decoder, JPEG Decoder, MP3 Encoder and Sample Rate Converter. Additionally, two non-streaming applications are also considered for this work. These are FFT and Romberg Integration from [34]. The voltage and frequency pairs for a core are dependent on the core and the system-on-chip (SoC) manufacturer and can be set for Linux operating system during the kernel configuration phase of the kernel build process. Since this work is simulation-based, the voltage frequency pairs are selected based on Texas Instrument SoC [35], which integrates ARM Cortex-A8 core. The specifications of this core are reported in Table II. Although these voltage-frequency pairs are assumed for simplicity, the proposed algorithm and the temperature model can be trivially applied to any architecture with any supported voltage-frequency pairs.

The bit energy \( (E_{bit}) \) for modeling communication energy of an application is calculated using the formulas provided in [33] for packet-based NoC with Batcher-Banyan switch fabric using 65nm technology parameters from [36]. The parameters used for computing MTTF are the same as in [2] [10] [12]. The scale parameter of each core is normalized so that its MTTF under idle (non-stressed) condition is 10 years. All algorithms are coded in C++ and used with SDF3 tool for throughput and schedule construction and HotSpot for temperature characterization. Furthermore, Matlab regression toolbox is used for modeling the spatial dependency.

A. Time Complexity

The time complexity of the reliability optimization algorithm is calculated as follows. There are \( |S_{app}| \cdot (N_c - N_c^{min} + 1) \) loops in the algorithm 1. In each loop, the algorithm executes the \( REOpt() \) and the \( MSDF^3 \) routines. The complexity of \( REOpt() \) (Algorithm 3) is computed as

\[
O(C_3) = \eta \cdot N_o \cdot N_c \cdot N_f \cdot O(MSDF^3)
\]
Fig. 9. Temperature variation of the proposed model.

\[ O(C_1) = |S_{app}| \cdot N_c \cdot [1 + \eta \cdot N_a \cdot N_c \cdot N_f] \cdot O(MSDF^3) \]  

\[ = |S_{app}| \cdot \eta \cdot N_a \cdot N_c^2 \cdot N_f \cdot O(MSDF^3) \]  

The \( MSDF^3 \) tool uses a simulation-based approach to determine the schedule from a given actor distribution. The execution time is reported in Table III.

Finally, the complexity of Algorithm 4 is calculated as follows. For every iteration of the outer loop (number of cores), sorting of MTTF is performed once followed by memory lookup. If the memory lookup time is assumed to be constant and there are \( n \) applications enabled simultaneously on \( N_c \) cores, every loop is executed in \( O(n \log n) \). The overall complexity of Algorithm 4 is therefore \( O(N_c \times n \log n) \). On the multiprocessor platform considered, this algorithm takes between 80-100μsec for two to six simultaneous applications on an architecture with nine homogeneous cores.

**B. Validation of the Temperature Model**

The temperature model in Equation 19 incorporates only the voltage and frequency of the one-hop neighbors with all other cores operating at the highest operating point of (1.35V, 1GHz). To determine the pessimism in this approach, Figure 9 plots the temperature variation obtained using the simplified model of Equation 19 in comparison with the temperature obtained using the HotSpot tool by varying the operating points of the other neighbors. The execution time of the synthetic task on an architecture with nine homogeneous cores.

**C. Comparison with Accurate Temperature Model**

Finally, we compare the proposed temperature model with the steady-state dynamic temperature profile (SSDTP) generated using the iterative technique of [11] and the steady-state temperature model of [10]. A synthetic SDFG is considered for this experiment with a throughput requirement of 80 iterations per second. This translates to a steady-state period of 12.5ms. This SDFG is executed on the multiprocessor system with 9 cores. The steady-state iteration of the SDFG corresponds to a period of 12ms. The power profile of the SDFG varies within iteration and this variable power profile is repeated every iteration. With such a variable power profile repeated periodically, the steady-state temperature is not constant but varies according to the periodic power pattern as shown in Figure 10 with the red dashed line obtained using the temperature model of [11]. For the same power profile, the result with the proposed model is shown in the same figure with black solid line. The mean temperature for this two temperature curves are 63.5°C and 66.1°C, respectively. The temperature model of [10] assumes a steady-state value for the duration of operation, which corresponds to the average power in this duration. This is shown with blue solid line in the figure and corresponds to a temperature of 75°C. (11.5°C difference from the average temperature of [11]). Thus, in comparison to the temperature model of [11], the proposed temperature model is more accurate than the model of [10].

A point to note here is that, although the proposed model results in an average temperature close to that obtained using the accurate model of [11], the thermal cycling is not captured accurately leading to a misprediction of the thermal cycling related MTTF. However, the advantage is its simple form (non iterative as opposed to the iterative technique of [11]), which
TABLE IV
IMPACT OF IGNORING THE TEMPERATURE TRANSIENT PHASE.

<table>
<thead>
<tr>
<th>Apps</th>
<th>MTTF using the model of [2]</th>
<th>MTTF using the model of [21]</th>
<th>MTTF using the proposed model</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFT</td>
<td>6.1</td>
<td>5.4</td>
<td>6.7</td>
</tr>
<tr>
<td>MPEG4</td>
<td>7.2</td>
<td>6.8</td>
<td>8.5</td>
</tr>
<tr>
<td>JPEG</td>
<td>8.6</td>
<td>9.4</td>
<td>9.6</td>
</tr>
<tr>
<td>MP3</td>
<td>6.4</td>
<td>6.1</td>
<td>7.5</td>
</tr>
<tr>
<td>SRC</td>
<td>7.9</td>
<td>8.7</td>
<td>8.7</td>
</tr>
<tr>
<td>synth16</td>
<td>6.8</td>
<td>6.0</td>
<td>6.8</td>
</tr>
</tbody>
</table>

can be included in the design space exploration especially for multi-application use-cases.

D. Impact of Temperature Misprediction

As established in Section II, some of the existing techniques ignore the transient phase of the temperature. This leads to an inaccuracy in the temperature prediction and a corresponding inaccuracy in the MTTF computation. Furthermore, ignoring the spatial dependency leads to temperature misprediction. To establish the importance of the transient phase and the spatial dependency of the temperature on the MTTF results, an experiment is conducted with six applications (five real-life and one synthetic) on the multiprocessor platform with nine cores. Table IV reports three MTTF values (in years): the MTTF obtained using the proposed technique with the temperature model of [2] that considers steady-state temperature phase only; the MTTF obtained using the proposed technique with the temperature model of [21] that considers the temporal dependency only; and the MTTF obtained using the proposed technique with the proposed temperature model.

For application FFT, the MTTF considering the proposed temperature model is 10% and 24% higher as compared to the MTTF considering the temperature model of [2] and [21], respectively. The MTTF improvement by ignoring the spatial dependency (column 3 vs column 4) is higher than the MTTF improvement ignoring the transient phase (column 2 vs column 4). A similar trend is observed for MP3 Decoder and H.264 Encoder. These results signify the importance of the spatial temperature component in the temperature estimation. A point to note here is that, the MTTF improvement by ignoring the spatial dependency is dependent on the size of the application executed on the platform. For JPEG application that uses only two cores of the architecture, the improvement is less than 3%. A similar trend is observed for Sample Rate Converter (SRC) application. Finally, as established in Section II, considering the steady-state temperature is accurate only if the execution times of the actors of an application are comparable to the time constant of the RC equivalent circuit. This is shown for the synthetic application synth16 (with 16 actors) in the table. The execution times of the actors are generated with a mean of 200s and standard deviation of 20s. As can be seen, the MTTF obtained using the proposed model and the model of [2] are the same. On average for all the applications considered (seven real-life and synthetic), including the six shown in the table, the proposed model improves MTTF by 8% as compared to the model in [2] and 15% as compared to that of [21].

To give further insight into the temperature misprediction considering the steady-state temperature model of [2], experiments are conducted with a set of synthetic SDFGs. The total number of the actors in these applications are fixed to 16 and the execution times of the actors are normally distributed with a varying mean value and a fixed standard deviation. The mean of the actor execution times is varied from 1s to 100s in steps of 10s. The MTTF obtained using the temperature model of [2] is normalized with respect to the MTTF obtained using the proposed model. This is shown in Figure 11. For small actor execution time, the MTTF using the proposed model is higher than that of [2] by 7%. As the mean execution time is increased, the two models become close and temperature difference is less than 0.01%.

Thus far, the validation of the proposed temperature model is presented. In the next few subsections, we present the results to validate the proposed approach.

E. MTTF Computation Considering Task Remapping

As established in Section III-D, modern multiprocessor systems support remapping of tasks (actors in the SDFG terminology) on the detection of faults. The MTTF for these systems need to be computed by considering task remapping as opposed to the native way of considering the time to the first fault. To determine the MTTF differences in the two computation techniques, experiments are conducted on the multiprocessor system with six cores and a set of six real-life
applications. This is shown in Figure 12(a). There are two bars for every application. The left bar is for the MTTF considering the first failure and the right one for MTTF considering remapping. As seen from the figure, the two MTTF values are similar for some of the applications such as FFT and MP3 decoder. For all the four other applications, the two MTTF values differ. On average for these applications, the MTTF improvement is 15%. To give more insight on the reason for such low MTTF difference for applications such as FFT as opposed to say, JPEG decoder, Figure 12(b) plots the mean and the standard deviation of the aging of the different cores for all the six applications. The standard deviation of the aging values is a measure of how much the aging of the individual cores differ from the mean value. A low standard deviation indicates a balanced situation with all cores suffering similar wear-out. On the other hand, a high standard deviation indicates some cores age faster than others. The standard deviation is normalized with respect to the mean value of the aging.

As seen from the figure, for applications such as FFT and MP3 Encoder, the standard deviation of the aging parameters is close to zero and thus the wear-out experienced in the cores due to these applications are similar. For these applications, the MTTF considering the first failure is similar (less than 0.5% lower on average) to the MTTF considering remapping. This is intuitive, because with all cores suffering similar wear-outs, the break point (time at which a core fails due to wear-out) for all the cores are similar and therefore remapping leads to an insignificant gain in lifetime. For all the other applications, the standard deviations are high, with some applications having standard deviation as high as 60% of the corresponding mean value. For these applications, the aging values are not balanced. Although a balanced aging leads to a higher overall MTTF, a further investigation into these applications reveals that the balanced aging mapping for these applications consumes higher energy and therefore the gradient-based heuristic selects the mapping with non-balanced aging, but with significantly lower energy consumption. For these applications, the MTTF computation considering remapping is higher by as much as 24% (average 10%) than the MTTF computation considering the time to the first failure (TTFF).

Finally, Table V reports the processor years considering task remapping for MPEG4 is 30% higher than the processor years considering TTFF. This improvement is due to the non-zero processor years with 5 and 4 active cores. This improvement signifies that, even after the first fault, the multiprocessor system can be exploited to deliver 30% of the performance delivered during the time to the first fault. A similar trend is observed for all the other applications in the table. On average, the processor years considering task remapping is 15% higher than the processor years considering the time to the first failure.

F. Reliability and Energy Improvement

Figure 13 plots the energy and reliability results of the proposed approach in comparison to the existing reliability-energy joint optimization technique of [10] for the six real-life applications. Furthermore, to determine the reliability benefit of the dynamic voltage and frequency scaling, these two techniques are compared with the highest MTTF technique of [4] (referred to as MMax), which determines MTTF by solving a convex optimization problem. These results are represented as three bars corresponding to each application. A point to note here is that all the application SDFGs are converted to homogeneous SDFGs (HSDFGs) before applying the techniques of [4] [10].

The following trends can be followed from the figure. The energy consumption using the proposed approach and the existing energy-reliability joint optimization technique of [10] are lower than the highest MTTF technique of [4] that does not consider dynamic voltage and frequency scaling. The MTTF obtained using these techniques are also higher than the MTTF of [4]. These results signify that by slowdown of the actor computation, the reliability can be improved significantly.

3The conversion of an SDFG to HSDFG is of exponential complexity and therefore the proposed technique is the first technique for reliability-energy-performance optimization for multimedia applications represented as SDFGs.
On average for all the applications considered, the existing optimization technique minimizes energy consumption by 10% with a corresponding reliability improvement of 26% as compared to the highest MTTF technique. A point to note here is that, this technique is based on sequential execution of applications and therefore the scope for actor slowdown is limited in order to satisfy the throughput requirement. The energy improvement in this technique is therefore not significant. The proposed technique achieves better results than this technique by minimizing energy consumption further by an average 15% and increasing lifetime by an additional 18%. In comparison to [4], the proposed technique minimizes energy consumption by 24% and improves lifetime by 47%. These improvement can be attributed to

- the proposed temperature model that considers transient and steady-state phases as opposed to considering the steady-state temperature only;
- the MTTF computation considering remapping; and
- the pipelined scheduling technique of the proposed approach as opposed to the sequential execution of [10].

G. Design Space Exploration Speed-up

To highlight the speedup achieved by using the proposed design space exploration heuristic to jointly optimize energy-reliability, Table VI reports its execution time in comparison with the convex optimization based technique of [4] and the simulated annealing based technique of [10]. The execution time are recorded by running synthetic SDFGs with varying number of actors on two multiprocessor systems – with four and six cores, respectively. The number of actors is limited to 8 as the convex optimization fails to provide results beyond 8 actors even for running more than 12 hours. The time reported in this table are the average results obtained by generating multiple SDFGs. For example, the execution time for 6 actors on 4 cores is the average time taken by the three techniques for 10 different synthetic SDFGs with 6 actors each. For fair comparison, the temperature pre-characterization step is omitted for all these techniques and the time reported are the time for the respective technique – convex solver for [4], simulated annealing for [10] and proposed heuristic of Algorithm 3. As seen from the table, for small number of actors the execution time of the convex solver is comparable to that of the simulated annealing (better for 4 actors on 4 cores). However, as the number of actors is increased, the simulated annealing outperforms the convex solver. In comparison to these two techniques, the proposed approach improves execution time significantly achieving benefits for small and large problem sizes. On average, the execution time using the proposed technique is 70% and 50% lower with respect to [4] and [10], respectively.

H. Use-case Optimization Result

Since this work is the first work on use-case level MTTF optimization, there is no reference for comparison. However, two standard strategies are developed to distribute the cores among concurrent applications in a use-case – throughput-based core distribution (TCD) and equal core distribution (ECD). For implementing these approaches, the cores of the architecture are first distributed to the applications based on the corresponding strategy (equally or in the ratio of the throughput). The optimization technique proposed in this paper is then applied on individual applications to determine their MTTF. The overall MTTF of the use-case is the minimum of the MTTFs of the concurrent applications. The MTTF thus obtained for a use-case using both these strategies are compared with the MTTF obtained using the proposed MTTF-based core distribution technique. To demonstrate the advantage of the proposed approach for use-case optimization, a set of six synthetic use-cases are generated. Four of these use-cases are composed of synthetic applications and the two others are composed of real-life applications. These use-cases are executed on a multiprocessor system with nine cores. Figure 14 plots the MTTF for the three approaches for these use-cases. The composition of each use-case is indicated in the label of the figure, where the applications with alphabets are the synthetic applications. For the use-case A-B, the MTTF obtained by distributing the cores equally is 4.6 years. The TCD achieves better results by distributing the cores in the ratio of throughput requirement. The improvement in this technique is 27%. The proposed technique improves this further by achieving 3% higher lifetime. To understand the reason behind this improvement, a simple example is provided.

Let us consider a multiprocessor system with four cores and a use-case with two applications – synthA and synthB, with the throughput requirement of synthA as three time that of the synthB. The minimum number of cores required to satisfy the throughput requirement of synthA and synthB are two and one, respectively. Furthermore, let synthB stresses the system more (with a higher temperature) than synthA due to the higher execution time of the actors of synthB. Clearly, distributing the cores to these applications as 3:1 will not be optimal for MTTF. This example motivates and proves the importance of considering MTTF while distributing the cores of the architecture. As seen from the figure, for some use-cases such as A-B and G-H, the improvements using the proposed technique are insignificant. For other use-cases such as E-F and SRC-FFT, the improvements are more than 20%. On average for all the six synthetic use-cases, the proposed technique improves MTTF by 10% as compared to TCD and 140% as compared to ECD.
VIII. CONCLUSIONS

In this work, a simplified temperature model is proposed that incorporates not only the dependency of the temperature of a core on its voltage and frequency of operation, but also the influence of the neighboring core’s temperature. The proposed model is based on off-line temperature characterization using the HotSpot tool. Based on this model, a gradient-based fast heuristic is proposed to determine the voltage and frequency of cores such that the energy consumption is minimized, simultaneously maximizing the system mean time to failure (MTTF). The lifetime reliability computation also considers task remapping, which is a common practice for modern multiprocessor systems. The proposed approach is validated experimentally on multiprocessor systems with different number of cores using a set of synthetic and real-life application SDFGs executed individually as well as concurrently. Results demonstrate that the proposed approach minimizes energy consumption by an average 24% and maximize lifetime by 47% as compared to the existing works. Additionally, the proposed MTTF-aware core distribution for concurrent applications results in more than 10% improvement in lifetime as compared to the performance-aware core distribution.

REFERENCES


