Abstract—This paper proposes a design-time (offline) analysis technique to determine application task mapping and scheduling on a multiprocessor system and the voltage and frequency levels of all cores (offline DVFS) that minimize application computation and communication energy, simultaneously minimizing processor aging. The proposed technique incorporates (1) the effect of the voltage and frequency on the temperature of a core; (2) the effect of neighboring cores’ voltage and frequency on the temperature (spatial effect); (3) pipelined execution and cyclic dependencies among tasks; and (4) the communication energy component which often constitutes a significant fraction of the total energy for multimedia applications. The temperature model proposed here can be easily integrated in the design space exploration for multiprocessor systems. Experiments conducted with MPEG-4 decoder on a real system demonstrate that the temperature model is within 5% of the actual temperature clearly demonstrating its accuracy. Further, the overall optimization technique achieves 40% savings in energy consumption with 6% increase in system lifetime.

I. INTRODUCTION

Lifetime reliability is a crucial design concern for modern multiprocessor systems-on-chip (MPSoCs) as escalating power densities and hence temperature variations continue to accelerate wear-outs. This has attracted significant attention both in industry and academia to investigate on system-level techniques such as application mapping and scheduling to mitigate wear-outs leading to an extended mean time to failure (MTTF) [1]–[4]. These studies assume a fixed voltage and frequency for the underlying processing elements (PEs). However, modern PEs support a wide range of voltages and frequencies, which are often exploited to meet performance requirements and to perform voltage and frequency scaling (DVFS) to minimize energy consumption [5]. This has motivated researchers in recent years to study the impact of voltage and frequency scaling on lifetime reliability [6]–[8].

The existing studies on the DVFS-aware reliability optimization suffer from the following limitations. First of all, most of the existing techniques include expensive (time consuming) temperature simulation in the design optimization loop thereby achieving exponential design space exploration time, limiting their applicability to small problem sizes (number of tasks and/or cores). Secondly, the cyclic dependency between power, temperature, voltage and time is over-simplified and/or the influence of neighboring cores’ temperature on the temperature of a core is not considered in the existing techniques leading to temperature overestimation by a minimum of 17% with a corresponding 20% overestimation of MTTF. Thirdly, pipelined execution is not considered in these techniques which is essential to guarantee throughput of multimedia applications in particular. The directed acyclic graph based energy and/or reliability aware mapping techniques fail for pipelined execution due to the complex nature of the scheduling. Finally, in all the existing research on energy-reliability joint optimization, only communication energy or computation energy is considered but not both.

This paper proposes a technique to address the limitations discussed above. Following are the key contributions.

- A fast design space exploration heuristic to determine the application task mapping and voltage and frequency of cores to minimize energy consumption while maximizing the reliability of a homogeneous multimedia MPSoC.
- Considering time dependency (temporal effect) and the neighboring cores’ temperature (spatial effect) in determining the temperature of a core.
- Consideration of cyclic graphs with pipelined execution, typical of multimedia applications.
- Consideration of both computation and communication energy components of total energy.

The rest of the paper is organized as follows. A brief introduction of the related works is presented in Section II. This is followed by the proposed temperature model and an introduction to Synchronous Data Flow Graphs (SDFGs) in Sections III and IV respectively. The problem formulation and the solution are discussed in Section V. Results are presented in Section VI and Section VII presents the conclusions.
The temperature of a core is related to its power dissipation according to the following equation [9].

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

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 which is dependent on voltage, frequency and temperature. Thus, there exists a cyclic dependency between power and temperature which exaggerates the complexity in the design space exploration process for temperature/reliability optimization. Several simplification techniques have been proposed in literature. A steady-state approximation is proposed in [1] [3] [6]–[8] [14] to solve this cyclic dependency. A piece-wise linear approximation is proposed in [15]. A linear programming approach is proposed in [16] [17]. However, as established in [18], these techniques are far from being accurate. An iterative approach with eigenvalue decomposition based solution of Equation 1 with condensed equation is proposed in [18]. The solution to the differential equation using this technique is

\[ T(t) = e^{\kappa t} T(0) + e^{\kappa t} \left( e^{\kappa t} - 1 \right) C^{-1} P(0) \] (2)

where \( \kappa = -C^{-1} G \) and \( T(0) \) is the initial temperature. Figure 1(a) shows a reference architecture with cores interconnected in a mesh based topology. The temperature of any core, say core \( c_i \), depends on

A.1 The time of execution of a task.
A.2 The power (and hence voltage) of core \( c_i \).
A.3 The temperature (and hence voltage) of cores surrounding \( c_i \).

In all prior studies, the time dependency (A.1 above) or the effect of voltage of the neighboring core on the temperature of a core (A.3 above) is ignored. To signify the importance of temperature underestimation (by ignoring component A.3), an experiment is conducted with core \( c_i \) as idle and varying the voltage of the one-hop and two-hops neighboring cores (identified in Figure 1(a) by cores \( c_{i-1} \) to \( c_{i+1} \)). Results are plotted in Figure 2 for some combinations of neighboring core activity. For simplicity, one voltage level of 1.2V is assumed for every core in the architecture and the temperature results are normalized with respect to the temperature obtained only with A.1 and A.2. As can be seen from the figure, the existing techniques can result in 17% to 45% underestimation of temperature, which accounts for a significant difference (20% to 50%) in reliability estimation.

Incorporating the voltages of the neighboring cores in Equation 1 is complicated and involves solving multi-dimension differential equations. The technique in [18] performs 4-7 simulations for each mapping to determine the temperature of different cores with \( 0.5^\circ C \) accuracy. To improve this accuracy, the number of simulations needs to be increased limiting its adaptability for design space exploration with large number of tasks and cores. Instead, the proposed approach builds on Equation 2 and incorporates A.3 above by fitting the temperature data of a core \( T_i \) considering external (neighboring cores) as well as self voltage effect to the Matlab curve fitting toolbox to derive a polynomial relationship of the form

\[ T_i(0) = f_1(V_i) + f_2(\{V_j \mid \forall e_j \in N(c_i)\}) \] (3)
where \( V_i \) is the voltage of core \( e_i \) and \( N(e_i) \) are the cores in the neighborhood of \( e_i \). Performing exhaustive temperature simulations for different voltage combinations of all neighbors (one-hop neighbors, two-hops neighbors etc.) is time consuming. A first order of approximation involves considering the voltages of only the immediate neighbors i.e. the east, west, north and south neighbors of a core referred as \( V_e, V_w, V_n \) and \( V_s \) respectively with all other neighbors set to operate at the highest operating voltage. The power traces for these neighbors are generated by running applications at the desired supply voltages. Figure 3 plots the temperature of core \( e_i \) where \( V_i \) is increased from 0.8V to 1.2V for few of these neighboring voltage combinations. The data obtained is fed to the Matlab curve fitting toolbox to derive the following relation between temperature (in Kelvin) and voltage (in Volts).

\[
T_i(0) = 91.52 X_e + 64.28 (X_e + X_w + X_n + X_s) + 30
\]

where the variables \( X_j \)'s for \( j = \{e, w, n, s\} \) are defined as

\[
X_j = \begin{cases} \frac{\alpha}{V_j} & \text{if } e_i \text{ is non-existent or idle} \\ \frac{\alpha}{V_j} & \text{otherwise} \end{cases}
\]

IV. SYNCHRONOUS DATA FLOW GRAPHS

Synchronous Data Flow Graphs (SDFGs) are often used for modeling modern DSP applications [10] 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. The nodes of an 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. Figure 1(b) shows an example of an SDF Graph. There are four actors in this graph. In the example, \( a_1 \) has an input rate of 3 and output rate of 4. An actor is called ready when it has sufficient input tokens on all its input edges and sufficient buffer space on all its output channels; an actor can only fire when it is ready. The edges may also contain initial tokens, indicated by bullets on the edges, as seen on the edge from actor \( a_2 \) to \( a_6 \) in Figure 1(b). The following definitions and lemmas are stated. For a detailed treatment and proofs, interested readers are urged to refer [19].

Definition 1: (SDFG) An SDFG is a directed graph \( G_{app} = (A, C) \) consisting of a finite set \( A \) of actors and a finite set \( C \subseteq Port^2 \) of channels. Each actor \( a_i \) is a tuple \( \langle n_i, \Gamma_i \rangle \), where \( n_i \) is the number of execution cycles of \( a_i \) and \( \Gamma_i \) is the set \( \{ \tau_{ij} \mid \forall j \} \), where \( \tau_{ij} \) represent the tokens communicated from actor \( a_i \) to actor \( a_j \). The source of channel \( \chi_{ij} \) is an output port of actor \( a_i \), the destination is an input port of actor \( a_j \).

Definition 2: (REPETITION VECTOR) Repetition Vector Rpt of an SDFG \( G_{app} = (A, C) \) is defined as the vector specifying the number of times actors in \( A \) are executed for one iteration of SDFG \( G_{app} \). For example, in Figure 1(b), \( Rpt([a_0, a_1, a_2, a_3]) = [1 1 1 2] \).

Definition 3: (APPLICATION PERIOD) Application Period \( Per(A) \) is defined as the time SDFG \( G_{app} = (A, C) \) takes to complete one iteration on average.

One interesting properties of SDFGs relevant to this paper is throughput which is defined as the inverse of the long term period, i.e. the average time needed for one iteration.

Lemma 1: For a consistent and strongly connected SDFG, the self-timed schedule consists of a transient phase followed by a periodic (steady-state) phase.

This paper focuses on streaming applications represented as SDFGs. However, the techniques proposed are generic and applicable to both SDFGs and DAGs. Sections requiring special treatment for either of them are appropriately highlighted.

V. PROBLEM FORMULATION AND SOLUTION

A. Energy Modeling of Application

The leakage power of a core consumed during the execution of an actor is given by the following formula [20].

\[
P_{\text{leak}} = N_{\text{gates}} V_I \left( A \tau e^{-\frac{\alpha t}{V+\beta}} + B e^{-\frac{\gamma t}{V+\delta}} \right)
\]

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 [20]).

The dynamic power of a circuit is given by Equation 7 where \( \alpha \) is the activity factor, \( \omega(V) \) is the frequency (voltage) of operation and \( C_{eff} \) is the effective load capacitance.

\[
P_{\text{dyn}} = \alpha \omega(V) C_{\text{eff}} V^2
\]

The dynamic energy of an SDFG is given by \( E_{\text{dyn}} = E_{\text{dyn}}^r + N_{\text{iter}} \cdot E_{\text{dyn}}^r \) where \( E_{\text{dyn}}^r \) is the actor dynamic energy in the transient phase of the schedule, \( E_{\text{dyn}}^r \) 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 (can be regarded as periodic decoding of every frame for a video application) and hence for all practical purposes, the dynamic energy of the steady state phase dominates over that in the transient phase. Denoting \( t_{\text{iter}} = \frac{\omega}{\alpha} \) as the execution time of the actor \( a_i \) operating at voltage-frequency pair \( (V_j, \omega_j) \), the dynamic energy consumption is given by Equation 8.

\[
e_{ij} = P \cdot t_{ij} \cdot \text{Rpt}[a_i] = \alpha \cdot C_{\text{eff}} V_j^2 n_i \cdot \text{Rpt}[a_i] \]

A variable \( x_{ij} \) is defined as follows

\[
x_{ij} = \begin{cases} 1 & \text{if actor } a_i \text{ is executed at frequency } \omega_j \\ 0 & \text{otherwise} \end{cases}
\]

The total energy of the application is given by

\[
E_{\text{comp}} = \sum_{a_i} \sum_{j} c_{ij} \cdot x_{ij} + \sum_{\forall a_i \in A} \sum_{j} P_{\text{leak}} \cdot t_{ij}
\]
B. Communication Energy Modeling of Applications

In [21], bit energy ($E_{bit}$) is defined as the energy consumed in transmitting one data bit through an NoC router and link.

$$E_{bit} = E_{Sil} + E_{exit}$$

where $E_{Sil}$ and $E_{exit}$ 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 $\omega_{ hops}(p, q)$ away is given by Equation V-B according to [22].

$$E_{bit}(p, q) = \omega_{ hops}(p, q) \cdot E_{Sil} + (\omega_{ hops}(p, q) - 1) \cdot E_{exit}$$

if $p \neq q$

otherwise

The total communication energy is given by Equation 12.

$$E_{comm} = \sum_{\forall a_i, a_j \in A} d_{ij} \cdot E_{bit}(\Phi(i), \Phi(j))$$

where $\Phi(i)$ and $\Phi(j)$ are the cores where actors $a_i$ and $a_j$ are mapped respectively and $d_{ij}$ is the data communicated from actor $a_i$ to $a_j$ and is given by $d_{ij} = Rpt[a_i] \cdot \tau_{ij} \cdot sze$, where $sze$ is the size of a token in bits.

C. MTTF Modeling of Architecture

The lifetime reliability of a core is given by $R(t) = e^{-At}$, where $A$ is the rate of aging of the core per iteration of the application graph and is given by (refer [1], [3], [23]).

$$A = \frac{1}{\omega} \sum_{\forall \omega} \Delta t_i$$

where $\omega$ is the period of the application graph, $\alpha(T)$ is the fault density (typically Weibull or Lognormal distribution) and $T_j$ is the average temperature in the interval $\Delta t_i$. The MTTF of core $p_j$ with reliability $R_j(t)$ is given by

$$MTTF_j = \int_{0}^{\infty} R_j(t) \cdot dt = \int_{0}^{\infty} e^{-t(A_j)^{\beta}} \cdot dt$$

The MTTF for a multi-core system with $\omega_{ arcs(G)}$ cores is determined by the minimum of the MTTFs of the constituent cores (similar to that used in [3], [4], [6], [8]). Throughout the rest of this paper, MTTF of an MPSoC platform refers to the minimum of the MTTFs of the different cores of the system.

$$MTTF = \min_{j} (MTTF_j)$$

The MTTF and energy are combined into a single metric.

$$Obj = \frac{MTTF}{E_{comp} + E_{comm}}$$

The optimization objective can be written as

Maximize $Obj$

Subject to

- The throughput requirement is satisfied.
- All control/data dependencies are satisfied.
- $MTTF \geq$ MPSoC MTTF constraint.

The objective function of the optimization problem is non-linear and therefore a gradient-based fast heuristic is proposed to solve the same. This is shown as pseudo-code in Algorithm 1. The algorithm starts from a starting mapping, schedule and throughput computed using $SDF^3$ tool of [24] (line 1). Subsequently, the algorithm moves every actor to every core in-order to determine a priority function which is defined as

$$P_t = \begin{cases} \frac{Obj_{1} - Obj}{Obj_{1}} & \text{if } T_t < T \\ \frac{Obj - Obj_{1}}{Obj_{1} - Obj} & \text{otherwise} \end{cases}$$

Here two cases are considered. If the throughput of the current move is lower than the original throughput, a gradient function is used to calculate its priority i.e. moves with the maximum increase of the objective function with the least throughput degradation are given higher priorities. In the second case, if the throughput is higher than the original throughput, higher priorities are given to moves with the largest increase in the objective function.

The algorithm remaps actor $a_i$ to a core $c_j$ at a frequency $\omega_k$ (lines 4-6). The Mapping is changed together with the execution time of $a_i$ (line 7). These information are fed to the modified $SDF^3$ tool to compute the throughput and schedule corresponding to a given mapping (line 8). The energy is computed using Equations 10 and 12. The MTTF is computed using Equation 15 using temperature-voltage relationship as established in Section VI. Once all the metrics are determined, the algorithm computes the priority function (line 10). If the same is greater than the best priority obtained thus far, the best values are updated (line 12). The algorithm continues until a move is found without violating the throughput requirement. When this happens, the algorithm terminates.

VI. RESULTS

Experiments are conducted with fifty synthetic and seven real-life multimedia benchmark SDFGs generated using the $SDF^3$ [24] tool. The number of actors in synthetic SDFGs range from nine to twenty-five. These encompass both computation and communication dominated applications. The seven real SDFGs are $H.263$ Encoder, $H.264$ Decoder, $H.264$ Encoder, MPEG4 Decoder, JPEG Decoder, MP3 Encoder and Sample Rate Converter. These applications are executed on an MPSoC architecture consisting of nine cores arranged in $3 \times 3$ mesh architecture. Five voltage-frequency pairs are assumed for each core. Although these parameters are assumed for simplicity, the algorithms can be trivially applied to any architecture with any supported frequencies. The bit energy ($E_{bit}$) for modeling communication energy of an application is calculated using expressions provided in [21] for packet-based NoC with Batcher-Banyan switch fabric using 65nm technology parameters from [25]. The parameters used for computing MTTF are same as [1] [3]. The scale parameter of each core is normalized so that its MTTF under idle (non-stressed) condition is 10 years. Algorithms developed in this paper are coded in C++ and used with $SDF^3$ tool.

Algorithm 1 Generate Mapping

Input: $G_{app}, G_{arc}$ and throughput constraint $T_c$

Output: Mapping which maximizes $Obj$

1: $[M, S, T] = SDFF^3(G_{app}, G_{arc})$

2: while true do

3: $P_t = 0$, $M_t = M$, $found = false$, Calculate $Obj$ using $S$

4: for all $a_i \in A$ do

5: for all $c_j \in G_{arc}$ do

6: if found then

7: $M_t = M_t \oplus a_i$

8: $[S_t, T_t] = SDFF^3(M_t, G_{app}, G_{arc})$

9: Calculate $Obj$ using $S$

10: Calculate the priority $P_t$

11: if $T_t > T_c$ and $P_t > P_i$ then

12: $P_i = P_t$, $M_i = M_t$, $found = true$

13: end if

14: end if

15: end for

16: end for

17: if found then

18: $M = M_t$ and $T = MSDFF^3(M, G_{app}, G_{arc})$

19: else

20: break

21: end if

22: end while

23: Return $M_{new} = M_t$
for throughput and schedule construction and HotSpot for temperature characterization. Further, Matlab curve fitting toolbox is used to establish the voltage/frequency-temperature relationship as well as the relationship of the neighboring cores’ voltage/frequency on the temperature of a core.

A. Validation of the Temperature Model

The temperature model in Equation 4 takes one hop neighbors into account i.e. the neighboring cores located at a maximum distance of one hop from the core \( e_i \). To determine the pessimism in the proposed temperature model, Figure 4 plots the temperature variation obtained using the simplified model of Equation 4 in comparison with the actual temperature obtained by varying the voltage levels of the other neighbors. Results for the one hop neighbors are obtained by varying the voltages of the cores located at a distance of one hop from \( e_i \) with all other neighbors set at idle voltage. Similarly, the results for two hops are obtained by varying the voltages of the cores located at one and two hop distances from \( e_i \) with other neighbors set at idle voltage. All temperature values are normalized with the temperature obtained using the model in Equation 4. Although the temperature model is characterized with 1.2V set on the non-nearest neighbor, the model gets more accurate as all the cores are operational at 1.2V.

Another important point to note is that, the proposed temperature model incorporates a linear dependency of temperature on the voltages. To determine the accuracy of this model, experiments are conducted using multi-threaded MPEG 4 decoder application to determine the temperature predicted using HotSpot. This is shown in Figure 5 for 600S of video decoding using 1 to 4 cores. Further, to determine the difference in the predicted temperature with the actual temperature, the same application is executed on Hardkernels Odroid-X embedded system with four ARM Cortex-A9 cores with temperature reading from the on-board thermal sensor. The architecture parameters (such as heat sink) for the HotSpot tool are specified to the best of authors’ understanding, similar to those of the Odroid architecture. As can be seen from the figure, the temperature predicted using the linear model and the HotSpot tool are similar for single core (Figure 5(a)). As more cores are used in the system, the HotSpot temperature prediction is higher than that of the one predicted using the model. For four cores, the temperature using the model is within 5% of the actual temperature results.

B. MTTF-Energy-Performance of the Proposed Technique

Figure 6 plots the MTTF, energy and performance (measured as throughput) of the proposed technique in comparison with the highest MTTF technique of [3] (referred as \( MTTF_{\text{Max}} \)), the MTTF and communication energy minimization technique of [8] (referred as \( MTTF_{\text{MaxCommMin}} \)) and the MTTF and computation energy minimization technique of [6] (referred as \( MTTF_{\text{MaxCompMin}} \)). The number of actors in the SDFGs are limited to 12 as the convex technique of [3] and the simulated annealing based technique of [6] fail to provide results for larger SDFGs. Further, all SDFGs are first converted to homogeneous SDFGs (HSDFGs) before applying the techniques of [8] [6].

There are a few trends that can be followed from this figure. First of all, for the computation dominated applications such as synth6, synth9 and synth12, the \( MTTF_{\text{MaxCompMin}} \)
TABLE I
SIGNIFICANCE OF RELIABILITY OVERESTIMATION

<table>
<thead>
<tr>
<th>Applications</th>
<th>MTTFMaxTT</th>
<th>Actual MTTF</th>
<th>MTTF using</th>
<th>Actual MTTF</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>synth6</td>
<td>8.9</td>
<td>7.3</td>
<td>7.7</td>
<td>4.8%</td>
<td></td>
</tr>
<tr>
<td>synth69</td>
<td>8.6</td>
<td>7.7</td>
<td>8.3</td>
<td>8.4%</td>
<td></td>
</tr>
<tr>
<td>synth12</td>
<td>8.2</td>
<td>7.3</td>
<td>7.7</td>
<td>8.4%</td>
<td></td>
</tr>
<tr>
<td>jpeg</td>
<td>8.8</td>
<td>7.4</td>
<td>7.7</td>
<td>3.4%</td>
<td></td>
</tr>
<tr>
<td>h264</td>
<td>8.6</td>
<td>7.3</td>
<td>8.0</td>
<td>10.1%</td>
<td></td>
</tr>
<tr>
<td>mpeg</td>
<td>9.2</td>
<td>7.0</td>
<td>7.3</td>
<td>2.9%</td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6.0%</td>
</tr>
</tbody>
</table>

TABLE II
DSE WITH VARYING ACTORS AND CORES

<table>
<thead>
<tr>
<th>Actors</th>
<th>DSE Time (minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>cores = 4</td>
</tr>
<tr>
<td>4</td>
<td>39</td>
</tr>
<tr>
<td>6</td>
<td>79</td>
</tr>
<tr>
<td>8</td>
<td>677</td>
</tr>
<tr>
<td>10</td>
<td>1503</td>
</tr>
</tbody>
</table>

C. Design Space Exploration Speedup
Table II reports the execution time of the proposed approach in comparison with the convex optimization and the simulated annealing based existing techniques as the number of actors is increased for two different architectures. The design space exploration time for [3] (and [6]) includes the execution time of the convex optimization (and simulated annealing). The execution time of the proposed approach includes the time for Algorithm 1. As can be seen, the proposed technique reduces the execution time by an average 70% and 50% with respect to [3] and [6] respectively.

VII. CONCLUSION AND FUTURE WORKS
This paper presents a technique to study the energy-reliability-performance trade-offs for multimedia applications modeled as synchronous data flow graphs. By pre-characterizing the temperature dependency on surrounding core voltages as well as the self voltage, the proposed approach achieves 50% speed-up as compared to the existing approaches. Further, temperature-aware optimization technique improves energy consumption by 40% with 6% increase in MTTF. In future works, heterogeneous architecture will be considered.

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

REFERENCES