Temperature Aware Energy-Reliability Trade-offs for Mapping of Throughput-Constrained Applications on Multimedia MPSoCs

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

***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 using the proposed 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 underestimation 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.

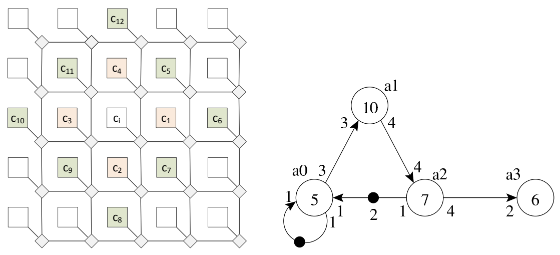
This paper proposes a technique to address the limitations discussed above. Following are the key contributions.

- 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.

HotSpot [9] tool is used to pre-characterize the relation between a processor voltage and temperature, and the effect of neighboring cores' temperature on the temperature of a core. A temperature model is proposed based on this and is validated against temperature obtained from sensors on a quad-core system by running MPEG-4 decoder. The temperature obtained using the on-board thermal sensors demonstrate that the proposed model is within 5% of the actual temperature. The model is used in the proposed design space exploration heuristic to estimate the temperature of a core to determine the impact on its aging and hence its reliability. The outcomes of the heuristic are (1) application task mapping and scheduling and (2) voltage and frequency of individual cores which are optimal in terms of reliability and energy consumption.

Experiments conducted with synthetic and real-life application graphs modeled as Synchronous Data Flow Graphs (SDFGs) [10] on homogeneous multiprocessor systems equipped with DVFS capabilities demonstrate that the proposed technique minimizes energy consumption by an average of 40% and MTTF by 6%. Furthermore, the proposed heuristic achieves a speedup of 50% as compared to the simulated-annealing and ILP-based existing techniques.

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.



(a) Architecture Model (b) Application Model

Fig. 1. Application and architecture example

II. Related Works

Design-time based task mapping and scheduling on a multi-processor platform has received a significant attention in recent past, focusing both on unrestricted optimization (best effort) as well as the energy and reliability driven restricted counterpart. A stop-go scheduling of task graph on DVS enabled MPSoC is proposed in [11]. This approach does not incorporate the impact of neighboring cores' temperature. Reliability-energy trade-offs are studied in [12] [13] by incorporating voltage/frequency in the transient fault probability. A fault-aware resource management is proposed in [7] dealing with *proactive* and *reactive* fault-tolerance with energy-minimization. None of these approaches are suitable for permanent faults which is the focus of this paper. Temperature-energy trade-offs are studied in [14], however core lifetime is not considered. Moreover, the proposed temperature model does not incorporate temperature of the neighboring cores, resulting in an underestimate of the temperature and a corresponding overestimate of the core lifetime. The only known approaches that maximizes (or satisfies) system lifetime together with energy minimization are the ones proposed in [6] [8]. In [6] only task computation energy is considered, while in [8] only task communication energy is considered. Moreover, the temperature is determined in a pre-characterization step considering different combinations of active tasks ignoring the power (and hence temperature) dependency on time (temporal effect). Another limitation of the proposed calibration step is that the number of thermal simulations grows exponentially with the number of tasks.

III. Proposed Temperature Model

The temperature of a core is related to its power dissipation according to the following equation [9].

$$C\frac{dT(t)}{dt} + G\left(T(t) - T_{amb}\right) = P(t) \tag{1}$$

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

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

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 $\boldsymbol{c}_i$ depends on

A.1 The time of execution of a task.
A.2 The power (and hence voltage) of core $\boldsymbol{c}_i$.
A.3 The temperature (and hence voltage) of cores surrounding $\boldsymbol{c}_i$.

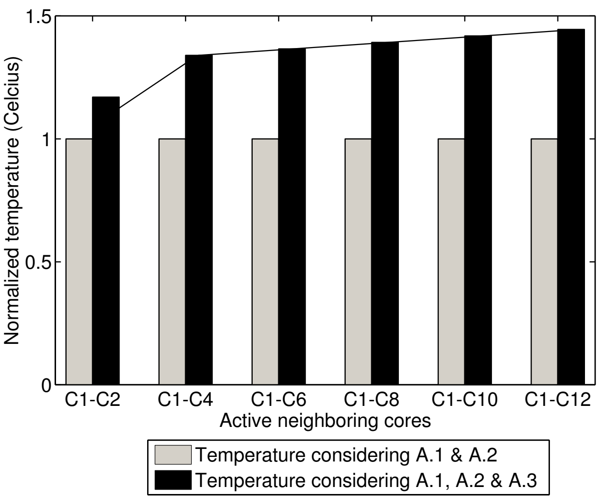


Fig. 2. Temperature underestimation by ignoring component A.3. The level $C1 - Cn$ indicates cores $\boldsymbol{c}_1, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_n$ active simultaneously.

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 $\boldsymbol{c}_i$ as idle and varying the voltage of the one-hop and two-hops neighboring cores (identified in Figure 1(a) by cores $\boldsymbol{c}_1$ to $\boldsymbol{c}_{15}$). 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^oC 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

$$T_i(0) = f_1(V_i) + f_2(\{V_j \mid \forall \boldsymbol{c}_j \in \aleph(\boldsymbol{c}_i)\}) \tag{3}$$

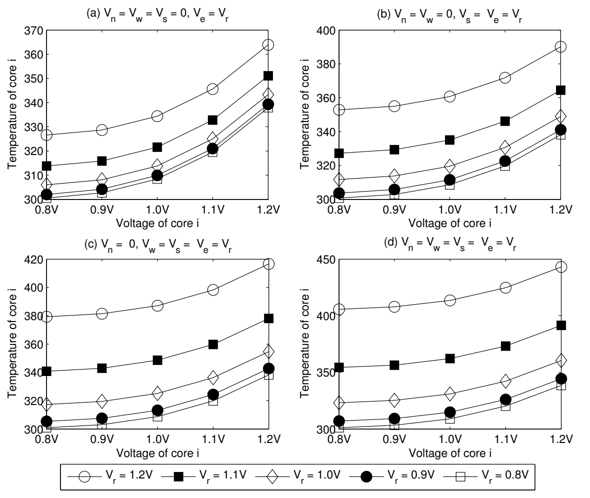


Fig. 3. Temperature (in Kelvin) Characterization

where V_i is the voltage of core $\boldsymbol{c}_i$ and $\aleph(\boldsymbol{c}_i)$ are the cores in the neighborhood of $\boldsymbol{c}_i$. 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 $\boldsymbol{c}_i$ as its voltage 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).

$$T_i(0) = 91.52X_i + 64.28(X_e + X_w + X_n + X_s) + 30 \tag{4}$$

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

$$X_j = \begin{cases} V_{idle} & \text{if } \boldsymbol{c}_j \text{ is nonexistent or idle} \\ V_j & \text{otherwise} \end{cases} \tag{5}$$

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 multiprocessor 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, $\boldsymbol{a}_1$ 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 $\boldsymbol{a}_2$ to $\boldsymbol{a}_0$ 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 $C \subseteq Ports^2$ of channels. Each actor $\mathbf{a}_i$ is a tuple $\langle n_i, \Gamma_i \rangle$, where n_i is the number of execution cycles of $\mathbf{a}_i$ and Γ_i is the set $\{\tau_{ij} \mid \forall j\}$, where τ_{ij} represent the tokens communicated from actor $\mathbf{a}_i$ to actor $\mathbf{a}_j$. The source of channel $ch_i^j \in C$ is an output port of actor $\mathbf{a}_i$, the destination is an input port of actor $\mathbf{a}_j$.*

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), $Rpt[\mathbf{a}_0\ \mathbf{a}_1\ \mathbf{a}_2\ \mathbf{a}_3]$ = [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].

$$P_{leak} = N_{gates} V I_0 \left[A T^2 e^{\frac{\alpha V + \beta}{T}} + B e^{\gamma V + \delta} \right] \tag{6}$$

where N_{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 α is the activity factor, ω (V) is the frequency (voltage) of operation and C_{eff} is the effective load capacitance.

$$P_{dyn} = \alpha * \omega * C_{eff} * V^2 \tag{7}$$

The dynamic energy of an SDFG is given by $E_{dyn} = E_{dyn}^{tr} + N_{iter} * E_{dyn}^{ss}$ where E_{dyn}^{tr} is the actor dynamic energy in the transient phase of the schedule, E_{dyn}^{ss} is the actor dynamic energy per iteration of the steady state phase and N_{iter} is the number of iterations of the steady state phase. Usually, the number of steady state iterations (i.e. N_{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 $t_{ij} = \frac{n_i}{\omega_j}$ as the execution time of the actor $\boldsymbol{a}_i$ operating at voltage-frequency pair (V_j, ω_j), the dynamic energy consumption is given by Equation 8.

$$e_{ij} = P * t_{ij} * Rpt[\boldsymbol{a}_i] = \alpha * C_{eff} * V_j^2 * n_i * Rpt[\boldsymbol{a}_i] \tag{8}$$

A variable x_{ij} is defined as follows

$$x_{ij} = \begin{cases} 1 & \text{if actor } \boldsymbol{a}_i \text{ is executed at frequency } \omega_j \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

The total energy of the application is given by

$$E_{comp} = \sum_i \sum_j e_{ij} * x_{ij} + \sum_{\forall \boldsymbol{a}_i \in A} \sum_j P_{leak} * t_{ij} \tag{10}$$

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.

$$E_{bit} = E_{S_{bit}} + E_{L_{bit}} \tag{11}$$

where $E_{S_{bit}}$ and $E_{L_{bit}}$ are the energy consumed in the switch and the link respectively. The energy per bit consumed in transferring data between cores $\boldsymbol{c}_p$ and $\boldsymbol{c}_q$, situated $n_{hops}(p,q)$ away is given by Equation V-B according to [22].

$$E_{bit}(p,q) = \begin{cases} n_{hops}(p,q) * E_{S_{bit}} + (n_{hops}(p,q) - 1) * E_{L_{bit}} & \text{if } p \neq q \\ 0 & \text{otherwise} \end{cases}$$

The total communication energy is given by Equation 12.

$$E_{comm} = \sum_{\forall \boldsymbol{a}_i, \boldsymbol{a}_j \in A} d_{ij} * E_{bit}(\Phi(i), \Phi(j)) \tag{12}$$

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

C. MTTF Modeling of Architecture

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

$$A = \frac{1}{t_p} \sum_i \frac{\Delta t_i}{\alpha(T_i)} \tag{13}$$

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

$$MTTF_j = \int_0^{\infty} R_j(t)dt = \int_0^{\infty} e^{-t(A_j)^{\beta}} dt \tag{14}$$

The MTTF for a multi-core system with $|G_{arc}|$ 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.

$$MTTF = \min_j \{MTTF_j\} \tag{15}$$

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

$$P_t = \begin{cases} \frac{Obj_t - Obj}{T - T_t} & \text{if } T_t < T \\ (Obj_t - Obj) & \text{otherwise} \end{cases} \tag{16}$$

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.

Algorithm 1 Generate Mapping

```
Input: $G_{app}$, $G_{arc}$ and throughput constraint $T_c$
Output: Mapping which maximizes $Obj$
 1: $[M \;\; S \;\; T] = SDF^3(G_{app}, G_{arc})$
 2: while true do
 3:     $P_b = 0$, $M_b = M$, $found = false$, Calculate $Obj$ using $S$
 4:     for all $\boldsymbol{a}_i \in A$ do
 5:         for all $\boldsymbol{c}_j \in G_{arc}$ do
 6:             for all $\omega_k$ supported do
 7:                 $M_t = M$ with $\boldsymbol{c}_j \leftarrow \boldsymbol{a}_i$
 8:                 $[S_t \;\; T_t] = MSDF^3(M_t, G_{app}, G_{arc})$
 9:                 Calculate $Obj_t$ using $S_t$
10:                 Calculate the priority $P_t$
11:                 if $T_t > T_c$ and $P_t > P_b$ then
12:                     $P_b = P_t$, $M_b = M_t$, $found = true$
13:                 end if
14:             end for
15:         end for
16:     end for
17:     if $found$ then
18:         $M = M_b$ and $T = MSDF^3(M_b, G_{app}, G_{arc})$
19:     else
20:         break
21:     end if
22: end while
23: Return $M_{n_{arc}} = M_t$
```

The algorithm remaps actor $\boldsymbol{a}_i$ to a core $\boldsymbol{c}_j$ at a frequency ω_k (lines 4-6). The Mapping is changed together with the execution time of $\boldsymbol{a}_i$ (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.263 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 3×3 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

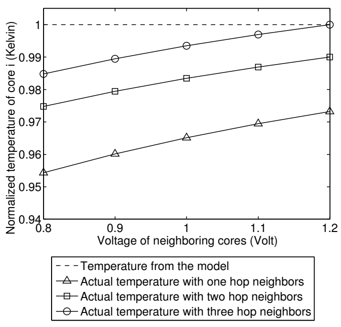


Fig. 4. Temperature variation of the proposed model

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 $\boldsymbol{c}_i$. 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 $\boldsymbol{c}_i$ 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 $\boldsymbol{c}_i$ 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

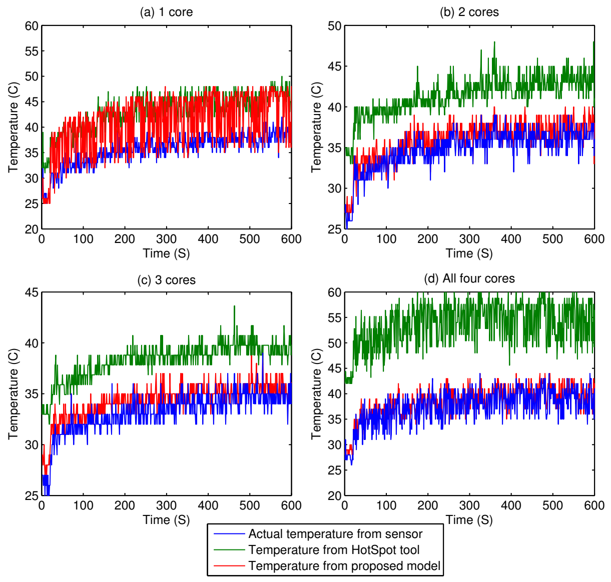


Fig. 5. Validation of the proposed model – MPEG 4 case study

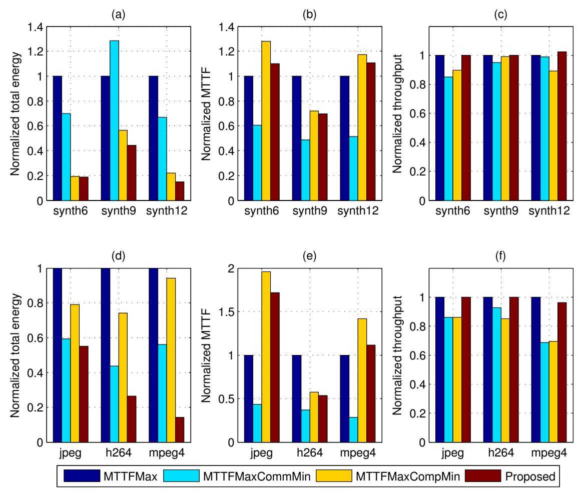


Fig. 6. Temperature-reliability-performance trade-offs

with the highest MTTF technique of [3] (referred as *MTTFMax*), the MTTF and communication energy minimization technique of [8] (referred as *MTTFMaxCommMin*) and the MTTF and computation energy minimization technique of [6] (referred as *MTTFMaxCompMin*). 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][1].

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 *MTTFMaxCompMin*

[1]The 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.

TABLE I
SIGNIFICANCE OF RELIABILITY OVERESTIMATION

Applications	**MTTF predicted using [6]**	**Actual MTTF using [6]**	**MTTF using *Proposed***	**Actual Improvement**
synth6	8.9	7.3	7.7	4.8%
synth9	8.6	7.7	8.3	8.4%
synth12	8.2	7.3	7.7	6.3%
jpeg	8.8	7.4	7.7	3.4%
h264	8.6	7.3	8.0	10.1%
mpeg	9.2	7.0	7.3	2.9%
Average				6.0%

TABLE II
DSE WITH VARYING ACTORS AND CORES

Actors	**DSE Time (in minutes)**					
	cores = 4			**cores =** 6		
	[3]	**[6]**	***Proposed***	**[3]**	**[6]**	***Proposed***
4	8	10	4	18	18	6
6	79	51	23	157	83	36
8	677	319	154	1013	524	274

technique achieves significant energy savings (on average 65% lower energy as compared to *MTTFMaxCommMin*). On the other end, the *MTTFMaxCommMin* achieves better result (on average 35% lower energy) than *MTTFMaxCompMin* for communication dominated applications such as *JPEG Decoder*, *H.264 Encoder* and *MPEG4 Decoder*. For both classes of application (computation and communication), the proposed technique achieves the least energy as both energy components are minimized simultaneously. On average for all applications considered, the proposed technique minimizes energy consumption by 70%, 55% and 40% with respect to the *MTTFMax*, *MTTFMaxCommMin* and *MTTFMaxCompMin* technique respectively. Secondly, the MTTF achieved using the proposed and the *MTTFMaxCompMin* are generally higher (better) than the other two techniques signifying the positive effect of voltage/frequency scaling on reliability. For some applications such as *JPEG Decoder*, the improvement is close to two fold. The MTTF of the proposed technique is lower than the *MTTFMaxCompMin* by only 10%. A point to note is that, the MTTF obtained for all techniques except the proposed one are an overestimate (due to the underestimation of the temperature). Table I reports the MTTF (in years) predicted using *MTTFMaxCompMin* [6] as compared to the actual MTTF (considering the neighboring temperature) and the MTTF obtained using the proposed technique. As can be seen from the table, the MTTF predicted using the existing technique is an overestimation by 20% (column 2 vs 3). The proposed technique increases lifetime by an average 6% as compared to the actual MTTF obtained using the existing techniques. Finally, the performance of the proposed and the *MTTFMax* are better than the other two techniques as both these techniques consider pipelined execution. A point to note here is that, the throughput requirement for all the real applications are relaxed as both *MTTFMaxCompMin* and *MTTFMaxCommMin* fail to satisfy the original throughput requirement for these applications. This once again demonstrate the advantage of the proposed approach in filling the gap existing in prior art for energy-reliability-performance trade-offs for multimedia application.

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 are 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% speedup 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

[1] L. Huang *et al.*, "Lifetime reliability-aware task allocation and scheduling for MPSoC platforms," in *DATE*, 2009.
[2] A. S. Hartman *et al.*, "A case for lifetime-aware task mapping in embedded chip multiprocessors," in *CODES+ISSS*, 2010.
[3] A. Das *et al.*, "Reliability-driven task mapping for lifetime extension of networks-on-chip based multiprocessor systems," in *DATE*, 2013.
[4] T. Chantem *et al.*, "Enhancing multicore reliability through wear compensation in online assignment and scheduling," in *DATE*, 2013.
[5] M. Schmitz *et al.*, "Energy-efficient mapping and scheduling for dvs enabled distributed embedded systems," in *DATE*, 2002.
[6] L. Huang *et al.*, "Energy-efficient task allocation and scheduling for multi-mode MPSoCs under lifetime reliability constraint," in *DATE*, 2010.
[7] C.-L. Chou *et al.*, "FARM: Fault-aware resource management in NoC-based multiprocessor platforms," in *DATE*, 2011.
[8] A. Das *et al.*, "Communication and migration energy aware design space exploration for multicore systems with intermittent faults," in *DATE*, 2013.
[9] K. Skadron *et al.*, "Temperature-aware microarchitecture: Modeling and implementation," *ACM TACO*, 2004.
[10] E. Lee *et al.*, "Synchronous data flow," *Proceedings of the IEEE*, 1987.
[11] P. Kumar *et al.*, "Thermally optimal stop-go scheduling of task graphs with real-time constraints," in *ASP-DAC*, 2011.
[12] P. Pop *et al.*, "Scheduling and voltage scaling for energy/reliability trade-offs in fault-tolerant time-triggered embedded systems," in *CODES+ISSS*, 2007.
[13] D. Zhu *et al.*, "Reliability-aware energy management for periodic real-time tasks," *IEEE Transactions on Computers*, 2009.
[14] S. Sharifi *et al.*, "Hybrid dynamic energy and thermal management in heterogeneous embedded multiprocessor socs," in *ASP-DAC*, 2010.
[15] V. Hanumaiah *et al.*, "Performance Optimal Online DVFS and Task Migration Techniques for Thermally Constrained Multi-Core Processors," *IEEE TCAD*, 2011.
[16] T. Chantem *et al.*, "Temperature-Aware Scheduling and Assignment for Hard Real-Time Applications on MPSoCs," *IEEE TVLSI*, 2011.
[17] A. K. Coskun *et al.*, "Temperature-aware mpsoc scheduling for reducing hot spots and gradients," in *ASP-DAC*, 2008.
[18] I. Ukhov *et al.*, "Steady-state dynamic temperature analysis and reliability optimization for embedded multiprocessor systems," in *DAC*, 2012.
[19] A. Ghamarian *et al.*, "Throughput analysis of synchronous data flow graphs," in *ACSD*, 2006.
[20] W. Liao *et al.*, "Temperature and supply voltage aware performance and power modeling at microarchitecture level," *IEEE TCAD*, 2005.
[21] T. Ye *et al.*, "Packetized on-chip interconnect communication analysis for MPSoC," in *DATE*, 2003.
[22] J. Hu *et al.*, "Energy-aware communication and task scheduling for network-on-chip architectures under real-time constraints," in *DATE*, 2004.
[23] J. Srinivasan *et al.*, "The case for lifetime reliability-aware microprocessors," in *ISCA*, 2004.
[24] S. Stuijk *et al.*, "SDF^3: SDF For Free," in *ACSD*, 2006. [Online]. Available: http://www.es.ele.tue.nl/sdf3.
[25] W. Zhao *et al.*, "Predictive technology model for nano-cmos design exploration," *ACM JETC*, 2007.