DRVS: Power-Efficient Reliability Management through Dynamic Redundancy and Voltage Scaling under Variations

Mohammad Salehi\textsuperscript{1}, Mohammad Khavari Tavana\textsuperscript{2}, Semeen Rehman\textsuperscript{3}, Florain Kriebel\textsuperscript{4}, Muhammad Shafique\textsuperscript{5}, Alireza Ejali\textsuperscript{6}, Jorg Henkel\textsuperscript{7}

\textsuperscript{1,6}Embedded Systems Research Laboratory (ESRLab), Sharif University of Technology, Tehran, Iran
\textsuperscript{2}Department of Electrical and Computer Engineering, George Mason University, Fairfax, Virginia, USA
\textsuperscript{1,3,4,5,7}Chair for Embedded Systems (CES), Karlsruhe Institute of Technology (KIT), Germany
\textsuperscript{1}mohammad.salehi@ce.sharif.edu, \textsuperscript{2}mkhavari@gmu.edu, \textsuperscript{3,4}{rehman.kriebel}@ira.uka.de, \textsuperscript{6}ejali@sharif.edu, \textsuperscript{5,7}{muhammad.shafique.henkel}@kit.edu

Abstract—Many-core processors facilitate coarse-grained reliability by exploiting available cores for redundant multithreading. However, ensuring high reliability with reduced power consumption necessitates joint considerations of variations in vulnerability, performance and power properties of software as well as the underlying hardware. In this paper, we propose a power-efficient reliability management system for many-core processors. It exploits various basic redundancy techniques (like, dual and triple modular redundancy) operating in different voltage-frequency levels, each offering distinct reliability, performance and power properties. Our system performs Dynamic Redundancy and Voltage Scaling (DRVS) considering process variations in hardware, and diversities in software vulnerability and execution time properties. Experiments show that DRVS system provides significant reliability improvements while providing up to 60% reduced power consumption compared to state-of-the-art techniques.

I. INTRODUCTION AND RELATED WORK

Many-core processors have become the mainstream for advanced computing systems [1][2][3]. However, scaling process technologies aggravates reliability of on-chip systems, e.g., due to soft errors that are transient faults due to high-energy particle strikes and manifest bit-flips in the underlying hardware [3]. Besides others, prominent techniques for mitigating soft errors in many-core processors are redundant multithreading (RMT) and process level redundancy [4][5]. Another issue related to aggressive technology scaling is manufacturing process variations, i.e., variations in frequency and leakage power of different cores on a chip or across different chips [6]. Fig. 1(a) illustrates that core-to-core frequency variations in an Intel’s 80-core test chip are 28% at 1.2V and 59% at 0.8V [6]. Since slow cores lead to performance degradation, these variations severely influence the system reliability and significantly hamper operating at the peak chip performance which might be crucial in timing-critical systems [7]. To mitigate the performance variations across different cores one solution is to increase the voltage and frequency of the slower cores in order to match up with that of the faster ones. However, slower cores may experience a high degree of process variations that cannot be compensated through the over clocking [6]. Therefore, exploiting multiple voltage-frequency levels allows more cores to work on their maximum frequency. This also enables power optimization potential across different cores in a chip [8]. Our analysis in Fig. 1 and Fig. 2 illustrates that, besides hardware-level variations, application programs (and even different functions of the same applications like DCT, SAD and SATD) also exhibit variations in (1) performance, i.e., execution time in terms of clock cycles; (2) vulnerability to soft errors due to varying data and control flow properties; and (3) power consumption due to diverse switching activity.

The goal of this paper is to synergistically exploit the above-mentioned hardware- and software-level variations in performance, power, and vulnerabilities to develop a power-efficient reliability management system for many-core processors. For this, managing core-to-core frequency to allow each core (or group of a few cores) to operate in its own maximum frequency is highly required to ensure satisfying timing constraints of different applications.

To mitigate process variation effects, the related works have employed mapping, scheduling and voltage/frequency assignment techniques [2][6][8]. Soft errors mitigation has been performed at program/software level (e.g., reliability-aware compilations to generate different iso-functional reliability-heterogeneous code versions [9] and at hardware level (e.g., vulnerability-driven Dual Modular Redundancy – DMR adaptation [10] and standby-sparing [11][12]). In multi-/many-core processors, soft error mitigation has been realized through employing idle cores for spatial and temporal redundancy [13]. Main works are: process-level [4] and core-level [14] redundancy, and chip-level redundant multithreading (CRT) [5]. These works assume that excessive cores are available for performing the same redundancy technique for all tasks (e.g., the use of DMR for all tasks [10]). This may not be applicable for massive multi-threading where several applications’ threads are competing for cores. Moreover, different applications may not even require the same redundancy technique to achieve a given reliability level due to their vulnerability variations (see Fig. 2 and Section I.A). Therefore, considering the same redundancy technique (like TMR or DMR or Standby Sparing) for all tasks may not lead to both inefficient core usage and high power...
consumption. Also, the above-mentioned works mainly consider soft errors and do not jointly account for hardware- and software-level variations that can lead to timing reliability degradation.

**In summary, to ensure a given reliability level with reduced power consumption requires synergistic adaptation of reliability techniques and power knobs (like voltage and frequency) under joint consideration of hardware- and software-level variations in vulnerability, performance and power properties.**

Before proceeding to our novel contributions, we present a detailed motivational analysis to illustrate the power and reliability design space that is leveraged by our proposed system.

A. **Motivational Analysis of Reliability-Power Tradeoffs**

**Software-Level Reliability Variations:** Fig. 2(a) shows that different applications and even their different functions exhibit distinct vulnerability to soft errors due to variations in the spatial and temporal vulnerabilities of different instructions (similar observation has also been reported by the work in [15]). Furthermore, the system-wide reliability of an application is a function of the software vulnerability, its execution time, and hardware-level error rate (see Eq. 2). Fig. 2(b) shows the variations in the system-wide reliability of different applications when executed using the maximum voltage-frequency level corresponding to a certain fault rate. Since different applications exhibit distinct reliability properties, they may achieve a given reliability level by using dissimilar redundancy techniques, thus facilitating power reductions for less-vulnerable applications.

**Reliability-Power Trade-off at Software- and Hardware-Levels:**

Fig. 3(a) shows that how hardware-level fault rate (given by Eq. 1) and power consumption vary at different voltage-frequency levels (Table I). This figure shows that lower voltage-frequency levels provide less power consumption but at the cost of a high fault rate. Furthermore, the application reliability depends upon both the hardware-level fault rate and the software vulnerability (see Eq. 2). Fig. 3(b) shows application reliability variations at different voltage-frequency levels. It illustrates that in order to achieve the same reliability level for different applications, it is not necessary to execute all the applications under the same voltage-frequency level. For instance, to achieve the reliability of 0.99999, the SATD application can be executed even on the lowest voltage-frequency level, i.e. [0.72V, 490MHz], whereas, DCT should be executed at [0.97V, 730MHz]. Therefore, variations in the reliability and power consumption at different voltage-frequency levels can be exploited for reliability-power management.

**Reliability-Power Trade-off for Various Redundancy Techniques:**

Fig. 4 shows the failure probability and power consumption for the DCT application when executed at different voltage-frequency levels and using different redundancy techniques, i.e., unprotected single execution (SE) that only employs an error detection mechanism like Argus [16] and redundant multi-threading in DMR and TMR modes [10][17]. The figure shows that a certain reliability level can be achieved at different voltage-frequency-redundancy combinations. Let us assume that the DCT application has a reliability requirement of a failure probability to be less than $10^{-8}$ (shown by the dotted horizontal line in Fig. 4(a)). In this case, SE mode cannot fulfill the reliability requirement, while TMR can satisfy the requirement at all voltage-frequency settings and DMR requires a setting of [0.97V, 370MHz] or higher. Now let us assume that DCT has a timing requirement that requires a frequency of 850MHz or greater (see vertical dotted line in Fig. 4(a)). In this case, both DMR and TMR can satisfy both the reliability and timing requirements at [1.1V, 850MHz] or higher. In summary, diversity in reliability and power consumption of different redundancy techniques at different voltage-frequency levels can be exploited to enable power-efficient reliability management at run time under constrained scenarios.

B. **Our Novel Contributions and Concept Overview**

We propose a novel Dynamic Redundancy and Voltage-Frequency Scaling (DRVS) system that enables power-efficient reliability by leveraging variations at software and hardware levels in terms of performance, power, and vulnerabilities considering different reliability mechanisms operating at different voltage-frequency (V-f) levels. Our DRVS system employs the following two key components (see overview in Fig. 5):

1) **A Reliability-Timing-Aware Scheduling and Mapping Scheme (Section III.B)** that dynamically enables a particular reliability mechanisms (out of multiple design-time options) for different concurrently executing applications considering their diverse performance, power, and reliability properties. Afterwards, it performs variation-aware core and V-f selection for redundant threads under timing requirements while minimizing the power consumption.

2) **A Dynamic Redundancy and Voltage-Frequency Scaling Scheme (Section II.C)** that updates the redundancy and V-f level at run time in case there is a change in the failure rate and/or in the number of available cores. In the former case, the redundancy and V-f levels for the task with low reliability are upgraded to provide high system reliability under high failure rates. In the latter case, the redundancy or V-f level is upgraded for the tasks with resource deficiency.

II. **SYSTEM MODELS**

**Hardware Architecture Model:** We focus on a many-core processor comprises $K$ homogeneous cores $\{C_1, C_2, \ldots, C_n\}$ each with its own data and instruction memories. The cores communicate through a hierarchical 4-ary tree structure to reduce the number of communication hops, like that considered in [2]. There exist multiple V-f operating levels with $V$ as the supply voltage and $f$ as the maximum possible frequency associated with the voltage $V$. Each V-f level contains one or more cores and the cores may have different maximum frequencies due to process variations. Following commercial and research trends of multiple V-F-islands (like in [8] and Intel SCC [18]), in the subsequent section, we consider that a group of cores work under a V-f level.

**Application Model:** An application consists of multiple tasks, such that, a task $r$ requires $w$ clock cycles for execution. The control and data dependencies among the tasks are modeled by a task graph $G=(T,E,H)$, where $T$ represents the task set and $E=\{e_{xy} \mid \tau_x, \tau_y, \epsilon T\}$ is the set of edges that represents the dependency among the tasks $(e_{xy})$ shows that $\tau_y$ is dependent to $\tau_x$ and cannot be executed before finishing $\tau_x$. $h_x, \epsilon H$ indicates the amount of data to be transferred between the tasks $\tau_x$ and $\tau_y$. 
Vulnerability Index (FVI) is a software reliability model called the comparison process [19].

As discussed in Section I.A, different application tasks exhibit different reliability characteristics. Similarly, the reliability of the redundant multi-threading in the TMR mode can be calculated using Eq. 3 where 

\[ R_{\text{TMR}}(R_1, R_2, R_3) = \mu_{\text{Voting}}[R_1 R_2 + R_1 (1-R_2) + R_2 (1-R_3) + R_3 (1-R_1)] \]  

(3)

Power Model: We consider that the power consumption of a core at V-f level is determined using Eq. 5 [11][20] where \( P_{\text{Staic}} \) and \( P_{\text{Dynamic}} \) are the static power (mainly consumed by sub-threshold leakage current \( I_{\text{sub}} \)) and dynamic power (mainly dissipated due to circuit switching activities), \( \alpha_{\text{act}} \) is the application activity factor, and \( C_L \) is the load capacitance.

\[ P(V, f) = P_{\text{Staic}} + P_{\text{Dynamic}} = I_{\text{sub}} V + \alpha_{\text{act}} C_L V^2 f \]  

(5)

III. OUR PROPOSED DRVS SYSTEM

As discussed in Section I.A, different application tasks exhibit different vulnerability, failure rates, and execution time properties. To increase the reliability of a task, one potential knob is to execute the task at a higher V-f level; see Fig. 4(a). Increasing the V-f level leads to reduced fault rates (Eq. 1), improved reliability (see Eq. 2) and reduced tasks execution but at the cost of increased power consumption (see Eq. 5). Another knob to increase reliability is the selection of an appropriate reliability mechanism (e.g., SE, DMR, or TMR), where a more robust reliability mechanism also corresponds to higher power consumption (see Fig. 4). Therefore, in order to achieve power-efficient reliability, we propose a Dynamic Redundancy and Voltage-Frequency Scaling (DRVS) system that selects a particular reliability mechanism (SE, DMR, or TMR) and V-f level considering variations at hardware and software levels under reliability and timings constraints of various concurrently executing tasks while minimizing the power consumption. Fig. 5 shows the overview of our DRVS system along with different inputs from both the hardware and software layers. Our DRVS system performs the following two key operations (see details in the subsequent sections).

A Reliability-Timing-Aware Scheduling and Mapping Scheme (Section IV.A, Algorithm 1): First, it selects an appropriate reliability mechanism and V-f level for a set of concurrently executing tasks under reliability and timing requirements while minimizing the power consumption. Afterwards, it maps and schedules these tasks in their respective selected reliability mechanisms on available free cores depending upon their reliability and timing requirements and underlying variations in performance and power properties. Our scheme first determines a near optimal mapping and scheduling solution (called base solution) depending upon the available free cores. Afterwards, at run time, depending upon releasing a core due to change in the system dynamics (e.g., a task finished its execution), our scheme updates the base solution using the following scheme.

A Dynamic Redundancy and Voltage-Frequency Scaling Scheme (Section IV.B, Algorithm 2): It may happen that due to resource limitation and set of constraints, the reliability mechanism and V-f levels that are assigned to the tasks during the previous step may not fully satisfy timing and reliability requirements for all the concurrently executing tasks. Therefore, when a core becomes free at run time, our scheme exploits it to upgrade the base solution with the aim of finding a better solution where timing and reliability for more tasks can be satisfied or a power-better wise solution can be found. Furthermore, in case of an increase in an application’s failure rate (e.g., due to increased deadline miss or other factors), our scheme can upgrade its reliability mechanism and V-f levels to achieve a more reliable solution. Vice versa, the reliability mechanism and V-f levels can be downgraded in case of a decrease in the application’s failure rates.

A. Problem Definition

We define the problem of the application scheduling and mapping on a many-core system when exploiting different reliability mechanisms and V-f levels for different application tasks as the Multi-Dimensional Bounded Knapsack Problem (BKP). The knapsack is the set of available cores \( C_{\text{free}} \). The objects in the knapsack are different reliability mechanisms, \( \tau = \{L_{SE}, L_{DMR}, L_{TMR}\} \), such that each mechanism \( L_i \) can be exploited for multiple tasks.

Bounds for Number of Copies for Different Object Types: In BKP each object is associated with a upper bound \( N_{\text{obj}}(L_i) \), which is a non-negative number. We assume that any core is allowed to form a reliability mechanism (i.e. SE, DMR or TMR) with any other core on demand (like that considered in [17]). Therefore, in our case, \( N_{\text{obj}}(L_i) \) is given as the maximum number of reliability mechanism \( L_i \) that can fit within the given \( C_{\text{free}} \).

\[ \forall L_i \in L, \quad N_{\text{obj}}(L_i) = C_{\text{free}} / L_i.Size \]  

(6)

Core Usage Constraint: The total number of used core, i.e., the sum of all assigned cores to the application tasks should be less than or equal to \( C_{\text{free}} \).

\[ \sum_{i=1}^{\tau C} C_i \leq C_{\text{free}} \]  

(7)

Reliability Constraint: The reliability mechanism \( \tau L \) and the cores V-f level (\( V_{f_\tau}, c_i \in \tau L \)) that are used for each application task should satisfy the reliability requirement \( R_{\text{req}} \).

\[ \forall \tau \in \tau, \quad R_{\text{\tau \_L}} \geq R_{\text{\tau \_req}} \]  

(8)

Timing Constraint: The selected cores and the cores V-f level (\( V_{f_\tau}, c_i \in \tau L \)) that are assigned to each application task should satisfy the timing requirement \( t_{\text{req}} \).

\[ \forall \tau \in \tau(\forall c_i \in \tau L, \max \{\text{ComOverhead}(c_i, \tau, \rho, h, c_i) + \tau \cdot w / f_i \} \leq t_{\text{req}}) \]  

(9)
Case 1: There is a reliability mechanism and V-f level that can satisfy both the timing and reliability requirements. To obtain this solution, at first we find the cores that can satisfy the timing requirement. Then, we check SE, DMR or TMR modes in a sequence to find the most appropriate reliability mechanism that can also satisfy the reliability requirement. Finally, the reliability mechanism with the lowest number of cores and also with the minimum power consumption is selected. Here, the task is scheduled and mapped such that both the timing and reliability requirements are satisfied and the power consumption is minimized. If there is no such solution case, we check the solution case 2.

Case 2: There is a reliability mechanism and V-f level that can only satisfy the reliability requirement. To obtain this solution, we find the reliability mechanisms (with a lower number of cores) that can meet the reliability requirement. Then, the cores with the maximum performance are selected and task mapping is performed in the maximum performance mode despite the fact that the timing requirement cannot be fully met, but yet providing minimal degradation. At run time, if a core becomes free, our scheme performs an upgrade for this task considering other competing applications.

Case 3: Best possible reliability mechanism and V-f level. In this case, if there are enough free cores for TMR mode, three cores with the maximum performance are assigned to the task. Otherwise, the remaining free cores (i.e. two or one core) are assigned to the task.

B. Reliability-Timing-Aware Scheduling and Mapping

Algorithm 1 shows the pseudo-code of our scheme. In the beginning of each iteration of the main while loop, the ready tasks are obtained (line 2). A ready task is a task that all its predecessor tasks in G are scheduled (e.g., a root task in the first iteration). Then, we estimate the reliability r, and execution time t, for each ready task τ on each free core c, and under each voltage-frequency level Vf. To do this, in the scheduling and mapping algorithm we consider the following three main solution cases.

**Optimization Goal:** We aim at selecting for each application task τ ∈ T the proper reliability mechanism and V-f level (τ,L and [Vi,fj], ci ∈ τ.L) such that the application power consumption is minimized while the application reliability and timing requirements (Eq. 8 and Eq. 9) are satisfied, i.e.,

\[
\text{minimize} \left( \sum_{\tau \in T} \sum_{c_i \in \tau.L} P(V_i,f_j) \right)
\]  

(10)

**Scheduling and Mapping Algorithm:** Optimally solving the multi-dimensional BKP problem is NP-hard, therefore, we develop a heuristic (Section III.B). To provide core usage efficiency, to leave more free cores for other demanding tasks and also to achieve power efficiency, at first we check whether a single core in SE mode can satisfy the task timing and reliability requirements. Otherwise, we use DMR or TMR. To do this, in the scheduling and mapping algorithm we consider the following three main solution cases.

1. **Case 1:** There is a reliability mechanism and V-f level that can satisfy both the timing and reliability requirements. To obtain this solution, at first we find the cores that can satisfy the timing requirement. Then, we check SE, DMR or TMR modes in a sequence to find the most appropriate reliability mechanism that can also satisfy the reliability requirement. Finally, the reliability mechanism with the lowest number of cores and also with the minimum power consumption is selected. Here, the task is scheduled and mapped such that both the timing and reliability requirements are satisfied and the power consumption is minimized. If there is no such solution case, we check the solution case 2.

2. **Case 2:** There is a reliability mechanism and V-f level that can only satisfy the reliability requirement. To obtain this solution, we find the reliability mechanisms (with a lower number of cores) that can meet the reliability requirement. Then, the cores with the maximum performance are selected and task mapping is performed in the maximum performance mode despite the fact that the timing requirement cannot be fully met, but yet providing minimal degradation. At run time, if a core becomes free, our scheme performs an upgrade for this task considering other competing applications.

3. **Case 3:** Best possible reliability mechanism and V-f level. In this case, if there are enough free cores for TMR mode, three cores with the maximum performance are assigned to the task. Otherwise, the remaining free cores (i.e. two or one core) are assigned to the task.
reliability across the free cores (line 12) and they are scheduled through lines 13-43. Here if no more free core is left, the algorithm returns and the application is not schedulable. Since the task with the lowest reliability is the most reliability-wise critical task, it is first selected (line 15). In line 16, we find the cores that can satisfy the timing requirement $t_{req}$. If there are such cores, we find cores in SE mode that can also satisfy the reliability requirement $R_{req}$. Then, the core with the minimum power consumption is selected (lines 18, 19). Otherwise, we consider a better reliability mechanism and evaluate whether the reliability requirement is satisfied. To do this, at first we examine dual cores in DMR mode which is more preferable than TMR from the core usage efficiency and power consumption points of view (lines 21, 22). If DMR cannot satisfy the reliability requirement, cores in TMR mode are examined and the cores with the minimum power consumption are selected (lines 24, 25). Here, if TMR can not satisfy the reliability requirement, three cores with the maximum reliability in TMR mode are assigned to the task (line 26). However, if there are not enough free cores for the TMR mode, remaining cores (i.e. two or one core) are assigned to the task (line 27). Until now, the algorithm tried to find a mapping for the task that satisfies both the timing and reliability requirement or satisfies the task timing requirement with the maximum possible reliability. In case no solution satisfies the timing requirements, our scheme selects a reliability mechanism that can satisfy the reliability requirement and the cores with the maximum performance are assigned to the task (lines 31-38). If required, three cores with the maximum performance in TMR mode are assigned to the task (line 39). Finally, if there are not enough free cores for the TMR mode, remaining cores (i.e. two or one core) are assigned to the task (line 40).

Note that in the above-discussed base solution it may happen that for some tasks, the timing or reliability requirements are violated. To address these issues, our scheme performs a dynamic redundancy and V-f scaling at run-time as discussed below in Section III.C.

C. Dynamic Redundancy and Voltage-Frequency Scaling

Algorithm 2 upgrades or downgrades the reliability mechanism and/or V-f level for different tasks, if required, depending upon the change in the system state, for instance, a core becomes free as a result of completion of a task’s execution or a failure rate of an application is increased or decreased. The algorithm selects the most reliability-wise critical task (line 3) and if its timing or reliability requirement is not satisfied or an error has occurred during the task execution, the algorithm tries to increase the task performance and reliability through scaling up the V-f level and reliability mechanism. To do this, at first it scales up the V-f level (lines 4-9) and if it is not possible to increase the V-f (when all associated cores to the task are performing on the maximum frequency $f_{max}$), it scales up the reliability mechanism as follows. It estimates the performance for all the free cores using the maximum communication time from the predecessor tasks plus the task execution time (lines 12-17). If there are some cores that satisfy the timing requirement, the core with the minimum power consumption is selected (line 20). Otherwise, the core with the maximum performance is selected (line 22).

IV. EVALUATIONS AND DISCUSSIONS

A. Experimental Setup and Processor Synthesis

We evaluate our DRVS system on various benchmark applications executed on a system-level many-core simulator, which is instrumented with precise power and performance characterization obtained through detailed ASIC synthesis and gate-level simulations.

### Table I. Power Consumption of LEON3 Processor Core.

<table>
<thead>
<tr>
<th>V-f Levels [Volt, MHz]</th>
<th>Power Consumption (mW)</th>
<th>Dynamic</th>
<th>Static</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1.23, 970]</td>
<td>641.39</td>
<td>16.47</td>
<td>657.86</td>
<td></td>
</tr>
<tr>
<td>[1.10, 850]</td>
<td>463.17</td>
<td>15.69</td>
<td>478.86</td>
<td></td>
</tr>
<tr>
<td>[0.97, 730]</td>
<td>336.81</td>
<td>15.08</td>
<td>351.89</td>
<td></td>
</tr>
<tr>
<td>[0.85, 650]</td>
<td>228.78</td>
<td>14.44</td>
<td>243.22</td>
<td></td>
</tr>
<tr>
<td>[0.72, 490]</td>
<td>140.62</td>
<td>13.34</td>
<td>153.96</td>
<td></td>
</tr>
</tbody>
</table>

**Fig. 7:** Reliability and power comparisons of our DRVS system with three state-of-the-art systems under low (L) and high (H) reliability and timing requirements ($R_{req}$ and $t_{req}$).

#### Power-Reliability Data:

The power model is based on Eq. 5 considering different V-f levels. First, we synthesized a VHDL design of LEON3 processor using the Synopsys Design Compiler and the TSMC 45nm low-power standard cell library (results are shown in Table I). To achieve signal activities for power estimation (i.e. $a$ in Eq. 5) we used ModelSim. In our experiments transient faults were generated by a Poisson process where to simulate transient fault rates in different voltages we used the model of Eq. 1 with $\lambda = 10^6$ and $\Lambda = 1/V$ [11]. Due to the stochastic nature of transient fault occurrences, test applications are executed for 1,000,000 times and the average results are reported.

**Generating Process Variation Maps:** To generate process variation maps we used the model in the work [21]. We considered 80×80 grid cells chips with standard deviation of process variation 10% of the nominal process parameter and correlation parameter 0.5 [22]. The effects of process variation on the maximum frequencies and leakage power were modelled through curve fitting on SPICE simulation points for a 13-stage ring oscillator with FO4 inverters of eight times of minimum size and 2-input NAND gate [23][24] (based on the 45nm metal gate/high-k Predictive Technology Model (PTM) [25]). The frequencies vary from 400 MHz to 1GHz for the lowest and highest voltage levels, with frequency binning in step of 10MHz. In experiments we generated several process variation maps with 32, 64 and 128 cores and average results are reported.

**Benchmark Application:** We used several combinations of real-world and synthetic applications to realize a wide range of competing application scenarios. The real-world applications are: DCT, FIR, SAD, SATD and different sorting algorithms. The synthetic task graphs were generated using TGFF [26]. In the experiments it is assumed that for the synthetic tasks the FIT varies between 0.1 and 0.3, the clock cycles vary between 2K and 20K and the activity factor ($\alpha$) varies between 0.01 and 0.5. These parameters vary between the minimum and maximum values of that for the real applications (see Section I.A).

B. Comparison with State-of-the-Art

#### Comparison Partners:

We compared our proposed DRVS system with three state-of-the-art systems: (1) dynamic DMR that activates/deactivates DMR mode for applications’ tasks according to the tasks vulnerability [10], (2) dynamic TMR where all tasks are executed in DMR mode and the TMR mode is activated when DMR cannot tolerate faults [17], (3) full TMR where tasks are all executed in TMR mode [5]. To provide a fair comparison we assumed that all the systems exploit DVS to achieve power reduction. In experiments we considered four combinations of low and high timing and...
reliability constraints. Under a lower timing requirement more time is available for tasks to be executed. Therefore, they can be executed under a lower V-f level to achieve power reduction. Similarly, under a lower reliability requirement lower redundancy levels can be exploited to meet the reliability requirement and to reduce power consumption. For comparison we used the following metrics (higher is better).

\[ \text{Reliability Profit} = \frac{\text{PoF}_{\text{DRVS}}}{\text{PoF}_{\text{M}}} \]

where PoF is the probability of failure and \( M \) (dynamic DMR [10], dynamic TMR [17], full TMR [5]). Fig. 7(a) shows reliability profit (in log scale). For instance, assuming that \( M \) provides reliability of 0.999 (i.e., PoF \( M = 10^{-9} \)) and DRVS provides reliability of 0.99999 (i.e., PoF \( \text{DRVS} = 10^{-5} \)), we have: Reliability Profit = 10\(^6\), and Reliability Profit (LogScale)=2. Fig. 7(b) shows the power reduction for DRVS over dynamic DMR [10], dynamic TMR [17], full TMR [5].

The following observations can be made from Fig. 7:

1) Under low reliability requirements (\( L_{R_{req}} \) in Fig. 7(a)), all the three systems provide higher reliability than DRVS (i.e., Reliability Profit > 0). This is because these systems use a fixed reliability mechanism to improve the reliability as much as possible without considering the reliability requirement. Therefore, these systems provide reliability higher than the required reliability, while the reliability of DRVS is high enough to meet \( R_{req} \) requirement. Since under low reliability requirements (\( L_{R_{req}} \)), our DRVS system exploits lower (but enough) redundancy and V-f levels, it achieves high power reduction compared to other systems (Fig. 7(b)). In this case, our DRVS system provides on average 28\%, 32\%, and 60\%, power reduction compared to the dynamic DMR [10], dynamic TMR [17], and full TMR [5], respectively.

2) Under high reliability requirements (\( H_{R_{req}} \) in Fig. 7(b)), DRVS provides higher reliability than the other three systems (i.e., Reliability Profit > 0). This is because DRVS exploits a proper redundancy level for each task to increase its reliability high enough to meet the reliability requirement. This results in an efficient use of free cores that provides an opportunity to increase the reliability for all tasks, and hence, an improved system-wide reliability is achieved. However, the other systems use the same redundancy level for all tasks, resulting in an imbalanced reliability for different tasks and a reduced system-wide reliability. Under high reliability requirements (\( H_{R_{req}} \)), like the other three systems, DRVS exploits high redundancy and V-f levels to meet the reliability requirement. Nonetheless, due to the efficient use of redundancy and V-f levels, DRVS yet achieves lower power consumption (Fig. 7(b)). In this case, DRVS provides on average 12\%, 9\%, and 21\%, power reduction compared to the dynamic DMR [10], dynamic TMR [17], full TMR [5], respectively.

3) Under high reliability requirements, when timing requirements are also high (i.e., \( H_{R_{req}} \) and \( H_{T_{req}} \)), DRVS achieves higher reliability profits over the other systems compared with the case that timing requirements are low (\( L_{T_{req}} \)). This is because, unlike DRVS, the other three systems do not consider timing requirements, and hence, encounter timing errors, resulting in high system-wide reliability degradation.

V. CONCLUSION

This paper presents a power-efficient reliability management system that accounts for both hardware-level process variations and software-level variability and performance. It employs a Dynamic Redundancy and Voltage-Frequency Scaling (DRVS) technique under reliability and timing constraints to achieve a good balance between reliability and power consumption. Compared to three different state-of-the-art techniques, DRVS achieves significant reliability improvements while providing 28\%-60\% power reduction. This paper demonstrates that joint consideration of variations at both hardware- and software-level provides opportunities for dynamic reliability management in low-power scenarios.

ACKNOWLEDGMENTS

This work is supported in parts by the German Research Foundation (DFG) as part of the priority program “Dependable Embedded Systems” (SP No. 1500 - sppl500.itec.kit.edu).

REFERENCES