dsReliM: Power-Constrained Reliability Management in Dark-Silicon Many-Core Chips under Process Variations
Mohammad Salehi1,2, Muhammad Shafique2, Florian Kriebel2, Semeen Rehman2, Mohammad Khavari Tavana3, Alireza Ejlali1, Jörg Henkel2
1ESRLab, Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
2Chair for Embedded Systems (CES), Karlsruhe Institute of Technology (KIT), Germany
3Department of Electrical and Computer Engineering, George Mason University, Fairfax, Virginia, USA
Corresponding Authors: salehi@ira.uka.de, muhammad.shafique@kit.edu

ABSTRACT
Due to the tight power envelope, in the future technology nodes it is envisaged that not all cores in a many-core chip can be simultaneously powered-on (at full performance level). The power-gated cores are referred to as Dark Silicon. At the same time, growing reliability issues due to process variations and soft errors challenge the cost-effective deployment of future technology nodes. This paper presents a reliability management system for Dark Silicon chips (dsReliM) that optimizes for reliability of on-chip systems while jointly accounting for soft errors, process variations and the thermal design power (TDP) constraint. Towards the TDP-constrained reliability optimization, dsReliM leverages multiple reliable application versions that can potentially execute on different cores with frequency variations and supporting different voltage-frequency levels, thus facilitating distinct power, reliability and performance tradeoffs at run time. Experiments show that our dsReliM system provides up to 20% reliability improvements under different TDP constraints when compared to a state-of-the-art technique. Also, compared to an ideal-case optimal solution, dsReliM deviates up to 2.5% in terms of reliability efficiency, but speeds up the reliability management decision time by a factor of up to 3100.

Keywords
Dark silicon, soft errors, process variation, many-core, reliability, constrained-optimization, power-efficiency.

1. INTRODUCTION
Continuous scaling of process technology has enabled high core integration where 100s of cores can be integrated onto a single many-core chip (1000s are envisaged by 2020 [1]). However, in such large-scale many-core chips the maximum number of cores that can be simultaneously powered-on (at the nominal voltage-frequency level) is constrained by the Thermal Design Power (TDP) [2][3]. The power-gated part of the chip is referred to as Dark Silicon [2][4][5]. The underlying reason is the exponential increase in the power density due to the deaccelerated scaling of threshold and supply voltages in the upcoming technology nodes and the exponential dependence of leakage power consumption on the threshold voltage [6][7][8]. Based on the technological data from Intel and ITRS [1], early analytical studies have predicted that more than 50% of a many-core chip may stay ‘dark’ at the nominal voltage-frequency level for 8 nm technology nodes [2].

Under a given TDP budget either less cores can operate at a high V-f level (i.e. at the full performance level) or relatively more cores can be powered on at a reduced voltage-frequency (V-f) level (i.e. at a lower performance level, sometimes also referred to as ‘Dim’ or ‘Gray’ Silicon ). Therefore, for a given TDP budget, the number of active cores will vary depending upon the frequency, leakage power, and operating voltage of the cores.

TDP denotes the maximum amount of power that can be dissipated for a given cooling technology to keep the temperature within safe operating limits [3]. It is important to note that, in case TDP is exceeded, the chip will generate more heat beyond the cooling capacity and elevated on-chip temperatures aggravate reliability threats like soft errors and aging [9][10].

High temperatures may accelerate the occurrence of permanent faults, for instance, due to time-dependent dielectric breakdown (TDBB) and electromigration [11], unless the chip is throttled down. Besides temperature-dependent increase in soft errors [10], rapidly changing power levels may lead to transient faults due to voltage noise [11]. It has been shown that circuit hardening cannot avoid the transient faults, e.g., only 30%-50% resilience improvement has been achieved for hardened flip-flops at sub-40nm in [12]. Therefore, state-of-the-art has focused on exploiting software and hardware level techniques to mitigate soft errors [8][13][14]. Since, these techniques are mainly based on redundancy and checking, they impose significant performance and power overheads to the system. For many-core on-chip systems, soft error mitigation is achieved through exploiting redundancy in different abstraction levels, e.g., exploiting idle cores for Redundant Multithreading (RMT) [13] and process- and core-level redundancy [14][15]. These techniques assume that excessive power budget is available for every application to exploit redundancy techniques. However, is no more the case in...
Figure 3: (a) Software vulnerability and performance (in terms of execution time); and (b) overall resulting system reliability (as per Eq. 2) for different compiled versions of different applications.

Figure 4: Reliability and execution time of four compiled versions for the ADPCM application under different voltage-frequency levels.

Dark Silicon chips. Mitigation of reliability threats under TDP constraints become even more challenging in the presence of process variations that result in chip-to-chip or core-to-core variations in leakage power and frequency [16]. Figure 1 shows our measured results for an example chip variation map illustrating core-to-core frequency and leakage power variations that need to be taken into consideration for power-efficient reliability optimization. Our analysis in Figure 2a illustrates that, besides the hardware-level variations, application softwares (and even different functions within the same applications like DCT and SAD) also exhibit the following two types of variations due to the diversity in their data and control flow properties: (1) variations in software vulnerability to soft errors due to varying vulnerability and masking probabilities of different instructions; and (2) variations in applications’ performance due to their varying execution time properties as a result of changing control flow. Since, the overall reliability of an application depends on both its software vulnerability and execution time (see Eq. 2), the above-discussed software-level variations manifest reliability variations for different applications even for the same underlying hardware (Figure 2b). Therefore, not all applications require the same degree of reliability improvement to maximize the overall system reliability and, hence, different reliability techniques with distinct power and performance properties can be leveraged to enable reliability optimization in Dark Silicon chips.

The objective of this paper is to cooperatively leverage the above-mentioned variations in performance, power, and vulnerabilities at both hardware and software levels in order to optimize the reliability of on-chip systems under TDP constraint.

1.1 Motivational Analysis: Exploring the Reliability and Performance Tradeoffs at Different V-f Levels

As extensively explored in the literature, a reliability-aware compiler (like in [17][18]) can be used to further expand the reliability-performance optimization space by generating multiple reliable code versions for a given application task. Different code versions of the same task implement the same functionality but they exhibit dissimilar reliability and performance properties. For instance, Figure 3 shows software vulnerability (in log scale), execution time (in terms of clock cycles) and overall reliability (see Eq. 2) of different compiled versions for three applications. Note, the reliability and execution time of an application also depend on the operating voltage and frequency of the underlying hardware. Figure 4 shows the reliability and execution time of four code versions for the ADPCM application under different voltage-frequency (V-f) levels.

Figure 4a illustrates that in order to achieve a given reliability level for the ADPCM application, different code versions of the application can be executed under different V-f levels. For instance, to meet the reliability constraint of 0.999 (shown by the dotted horizontal line in Figure 4a), the code version cv4 can be executed at a V-f level of [0.85Volt, 650MHz], whereas, the other three code versions should be executed at a V-f level of [0.97Volt, 730MHz] or higher. Let us assume that ADPCM has a timing constraint to finish its execution within 5ms (shown by the horizontal dotted line in Figure 4b). In this case, the code version cv2 can be executed at a V-f level of [0.85Volt, 650MHz], cv1 can be executed at a V-f level of [0.97Volt, 730MHz], whereas, the other two versions should be executed at a V-f level of [1.1Volt, 850MHz] or higher. Now let us assume that the underlying core has a TDP constraint that requires a V-f level of [0.85Volt, 650MHz] or less (the TDP1 constraint in Figure 4a). In this case, to satisfy both the reliability and TDP1 constraints, we need to select the code version cv4 (see Figure 4a). However, if a timing constraint has to be satisfied under the given TDP1 constraint, the code version cv2 would be selected (see Figure 4b). As another example let us assume the TDP2 constraint in Figure 4a (i.e. the core requires a V-f level of [0.97Volt, 730MHz] or less). In this case, the code version cv1 is the best choice since it can satisfy both the reliability and timing constraints of 0.999 and 5ms, while satisfying TDP2.

1 Also an 80-core test chip by Intel demonstrates frequency variations of 28% at 1.2V and 59% at 0.8V [16].

1.2 Concept Overview and Our Novel Contributions

To address the above-discussed challenges, we propose a novel Reliability Management system for Dark Silicon chips (dsReliM, Figure 5) that accounts for both hardware and software-level variations in performance, power, and reliability properties.
order to maximize the total system reliability, dsReliM performs
the following three key operations for different concurrently
executing applications at run time under the TDP and
application’s timing constraints:

1) Selecting an appropriate reliable code version for each
application considering their diverse vulnerability, execution
time, and power properties (note that multiple compiled
versions are given as an input to our dsReliM system);

2) Mapping of the selected code versions to appropriate cores
considering underlying frequency and leakage power
variations; and

3) At the same time, determining the voltage-frequency (V-f)
levels such that the mapped tasks meet their timing
constraints while keeping the total power consumption under
the TDP cap.

In order to achieve the above functionality in the dsReliM system,
we perform the following key steps.

1) Formal Problem Modeling: we model the problems of
reliability-driven code version, V-f level allocation, and task
mapping under TDP and timing constraints as a constrained
0-1 integer linear program (ILP). This model can be scaled
to arbitrary number of cores, V-f levels, application tasks and
code versions for each task.

2) Optimization Heuristics: we propose a computationally
lightweight yet efficient heuristic-based algorithm for solving
the 0-1 ILP reliability management problem at run time. We
evaluate the accuracy and runtime efficiency of our heuristics
by comparing it to an optimal solution.

Experimental results in Section 4 illustrate that our dsReliM
system provides reliability improvements of up to 19% compared
to state-of-the-art under different TDP constraints.

Compared to an optimal solution, it deviates up to 2.5% in terms
of reliability efficiency, but speeds up the reliability management
decision time by a factor of up to 3100 which makes it a feasible
solution for run time reliability management.

2. SYSTEM MODEL

2.1 Hardware Architecture Model

We consider the case of heterogeneous many-core processors
comprising of K cores \{C_{1}, C_{2}, ..., C_{K}\}. The cores can operate at
multiple V-f levels. For each voltage level \(V\), there exists a
maximum possible frequency \(f_{\text{max}}\). The maximum frequency and
leakage power consumption of different cores at each voltage
level may vary due to process variations. Considering Dark
Silicon chips, it is assumed that during each operating system
(OS) scheduling epoch only a subset of cores can be powered-on
to meet the chip TDP constraint. Also, the cores with higher
nominal V-f levels have higher power consumption when
executing identical applications. The overall performance of a
core with higher V-f levels is also expected to be higher than that
of a core with lower V-f levels.

2.2 Application Model

We consider applications consisting of multiple tasks. Each
application is represented by a task graph \(G=(T,E,H)\), where \(T\) is
the set of tasks and possible dependencies among the tasks are
represented by a set of edges \(E=\{e_{ij} | \tau_{ij}, r_{ij} \in T \} \) \(e_{ij}=1\) when \(r_{ij}\) is
dependent to \(\tau_{ij}\) and cannot start before finishing \(\tau_{ij}\). The amount
of data that is sent from \(\tau_{ij}\) to \(\tau_{ij}\) is determined by \(h_{ij} \in H\). In our
system, an existing reliability-aware compiler (like in [17][18]) is
exploited to create multiple reliable code versions for each
application task. Each code version requires \(w\) clock cycles for
execution. Also, different code versions of a task have different
vulnerability to soft errors since they differ in the spatial and
temporal vulnerabilities of different instructions [17][18].

2.3 Reliability Models

We consider transient faults, i.e. single- and multiple-bit upsets in
the underlying hardware. Such transient faults occurrences are
typically assumed to follow a Poisson process with the rate \(\lambda\) [19].

The fault rate varies exponentially with the supply voltage \(V\)
changes [20]. Therefore, the raw fault rate \(\lambda(V)\) corresponding to
the supply voltage \(V\) can be written as Eq. 1.

\[
\lambda(V) = \lambda_{0} 10^{\frac{V_{\text{max}}-V}{\Delta}} \tag{1}
\]

\(\lambda_{0}\) is the fault rate corresponding to the maximum voltage
\((V=V_{\text{max}})\) and \(\Delta\) is a parameter that determines the amount of
increase in fault rate when the voltage decreases by one level. A
transient fault in the underlying hardware may finally result in a
software failure. To measure the software failures due to transient
faults, we use a state-of-the-art software reliability model called the
Function Vulnerability Index (FVI) [17] that measures the
software failure probability and accounts for both spatial and
temporal vulnerabilities of different instructions (see details in
[17]). The software failure rate due to transient faults can be
modeled as \(\lambda(V) \times FVI\). Note that, our dsReliM system does not
depend on any dedicated software reliability model and any other
software reliability model can be used in dsReliM. According
to [19][21], the functional reliability \(FR\) of a software program
(i.e. probability of failure-free execution) under the software
failure rate \(\lambda(V) \times FVI\) can be written as Eq. 2.

\[
FR = \int_{V_{\text{min}}}^{V_{\text{max}}} \frac{1}{\lambda(V) \times FVI} dV \tag{2}
\]
FR(FVI, w, V, f) = e^{-\lambda(V_f + FVIx^2)}

w is the number of clock cycles that are needed for execution and w/f is the program execution time under the frequency f. In order to jointly consider the functional reliability FR and timing reliability TR (i.e., probability of meeting deadlines) we use the Functional-Timing Reliability model of Eq. 3 (like that is considered in [22]). In Eq.3, 0 ≤ α ≤ 1 is a user-defined parameter to determine the priority of functional or timing reliability (e.g., lower values for α represent more timing-critical systems).

\[ R = \alpha FR + (1 - \alpha) TR \]

### 2.4 Power Model

We consider that the total power consumption of a core is composed of static and dynamic power consumption. Static power \( P_{\text{Static}} \) is mainly dissipated due to sub-threshold leakage current and varies exponentially with the threshold voltage \( V_{th} \). Dynamic power \( P_{\text{Dynamic}} \) is mainly consumed due to the circuit switching activities. The total power consumption of a core at a V-f level and threshold voltage \( V_{th} \) can be written as Eq. 4 [11][19].

\[ P(V, f) = P_{\text{Static}} + P_{\text{Dynamic}} = I_d B V + a_{\text{sfa}} C_{m} V^2 f \]

\( I_d \) and \( \eta \) are technology parameters, \( V_T \) is the thermal voltage, \( a_{\text{sfa}} \) is the software activity factor, and \( C_{m} \) is the average switched capacitance.

### 3. OUR PROPOSED dsReliM SYSTEM

As discussed in Section 1.1, executing applications at a higher V-f level provides lower execution time and fault rate, resulting higher system-wide reliability; see Figure 4a. However, the power consumption at a high V-f level may exceed the TDP constraint of the underlying Dark Silicon chip. To decrease the power consumption, an effective way is to reduce the operating V-f level, e.g., through Dynamic Voltage Scaling (DVS) [23][24]. However, decreasing the V-f level leads to increased applications’ execution time that may result in performance degradation and timing errors. It is also discussed in Section 1.1 that different compiled versions for an application task exhibit different vulnerability and execution time properties. Therefore, the variations in vulnerability and execution time of different compiled versions for each task along with variations in reliability, power and performance at different V-f levels can be exploited for power-reliability optimization. We propose a Reliability Management system for Dark Silicon chips (dsReliM) that performs reliability-driven code version selection along with tasks mapping and V-f level allocation under process variations and TDP constraint.

#### 3.1 Problem Formulation

We use the following notation to represent the system reliability and power consumption, V-f level and code version assignments, and task-to-core mapping. In this formulation, \( n \) is the number of ready tasks, \( m \) is the number of available core versions for each task, \( c \) is the number of free cores, and \( v \) is the number of available V-f levels for each core:

- The system reliability is represented by the matrix \( R_{\text{c, m, c, v}} \), in which each element \( R_{ik} \) denotes the reliability of the task \( i \) when the code version \( j \) of the task is executed on the core \( k \) under the V-f level \( l \).
- The total power consumption is represented by the matrix \( P_{\text{c, m, c, v}} \), in which each element \( P_{ik} \) denotes the power consumption for the task \( j \) when the code version \( j \) of the task is executed on the core \( k \) under the V-f level \( l \).
- The code version and V-f level assignments and task-to-core mapping are represented by the matrix \( X_{ik} \) for the task \( i \) is mapped to the core \( k \) and is executed under the V-f level \( l \) if and only if \( X_{ikl} = 1 \).

The goal of the dsReliM system is to maximize reliability while keeping total power consumption under a given TDP constraint and meeting tasks timing constraints (deadlines). As power-reliability optimization knob, we leverage multiple reliable code versions for each application task that can potentially execute on different cores with frequency variations and supporting different V-f levels, each introducing distinct power, reliability and execution time properties. We formulate the problem as a constrained 0-1 integer linear program (ILP).

#### Optimization Goal: Maximize system reliability defined by the successful execution of all tasks.

\[ \text{maximize} \prod_{i, j, k, l} X_{ikl} R_{ikl} \]

Since this is a 0-1 assignment problem, we have:

\[ \forall i, j, k, l: X_{ikl} \in [0, 1] \]

#### Chip Power Constraint: Total power consumption, i.e. the sum of power of all underlying cores should be less than the chip TDP constraint.

\[ \sum_{i, j, k, l} X_{ikl} P_{ikl} \leq P_{\text{TDP, chip}} \]

#### Cores Power Constraint: Power consumption of each underlying core should be less than the core TDP constraint.

\[ X_{ikl} P_{ikl} \leq P_{\text{TDP, core}} \]

#### Tasks Timing Constraint: The execution time \( w_{j} / f_{ikl} \) for a code version \( j \) of a task \( i \) (with \( w_{j} \) clock cycles) on the core \( k \) and at the V-f level \( l \) should not exceed the task timing constraint (defined by the deadline).

\[ X_{ikl} w_{j} \leq \text{deadline}_{i} \]

#### Code Version Constraint: At each scheduling epoch, for each task only one code version can be used (the code version does not change during the task execution).

\[ \forall i, k, l: \sum_{j} X_{ikl} = 1 \]

#### Core Assignment Constraint: Each task can be only mapped to a single core.

\[ \forall i, j, l: \sum_{k} X_{ikl} = 1 \]

#### V-f Levels Assignment Constraint: Each core can be only run under a single V-f level (the V-f level does not change during the task execution).

\[ \forall i, j, k: \sum_{l} X_{ikl} \leq 1 \]
3.2 Proposed Solution

The formulated problem in Section 3.1 can be efficiently solved by existing ILP solvers (e.g., [25]). However, a 0-1 ILP problem is usually classified as a NP-complete problem [26] and hence ILP solvers typically employ branch-and-bound techniques [27] to search for the optimum solution. Therefore, their runtime complexity may increase exponentially with the increase of problem size, e.g., with the number of tasks, code versions, cores, and V-f levels in the case of our problem. Therefore, ILP solvers cannot be employed in online scenarios (e.g., when the ready and V-f levels in the case of our problem. Therefore, ILP solvers are usually classified as an NP-complete problem [26] and hence ILP solvers typically employ branch-and-bound techniques [27] to search for the optimum solution. Therefore, their runtime complexity may increase exponentially with the increase of problem size, e.g., with the number of tasks, code versions, cores, and V-f levels in the case of our problem. Therefore, ILP solvers cannot be employed in online scenarios (e.g., when the ready and core tasks and free cores are determined at run-time and a dynamic mapping is required).

To provide an effective run-time solution for the problem we develop a heuristic-based algorithm (Algorithm 1). The main idea is to first aggressively select software code versions and underlying cores and V-f levels such that highest possible reliability is achieved. Then, beginning from the core with the highest power consumption, decrease the V-f level to reduce the total power consumption until the point that the chip TDP constraint is met. Considering tasks timing constraints, when reducing the V-f level may cause a timing error (deadline miss), the code version is replaced with a version with less execution time. Figure 6 shows how different code versions and V-f levels can be exploited for reliability optimization under timing and TDP constraints. Suppose that for an application task we have different pre-compiled code versions (cv1, cv2, cv3,...) with different reliability and execution time properties. To achieve maximum reliability without considering the timing and TDP constraints, the code version with the highest reliability (i.e. cv1 in Figure 6) is selected to be executed at the maximum V-f level (V-fmax). In Figure 6 we assume without loss of generality that the execution time of cv1 at V-fmax is less than the timing constraint (denoted by deadline). However, let us consider that the power consumption at V-fmax exceeds the TDP constraint. In this case, the V-f level is scaled down until the TDP constraint is met. Suppose that after reducing the V-f level to a lower level the task execution time increases beyond the task deadline (the task timing constraint is missed). Therefore, we change the software code to a version that can meet the deadline. If there exist some code versions that meet the deadline, the code version that provides the maximum execution time reduction with minimum reliability loss is selected (i.e. code version with maximum time/reliability). However, if no code version can meet the task deadline, the code version with the minimum execution time is selected to provide graceful degradation. After changing the code version, we continue scaling down V-f level and updating the code version until the TDP constraint is met.

Our dsReliM system gets the set of ready tasks, the set of free cores, the core-to-core process variation map, and available V-f levels for each core. At design time, we select the appropriate code versions for each task amongst different compiled codes and use them at run time by Algorithm 1. At design time, for each application task, at first the code version with the highest reliability is determined. Then, among the other code versions, we select those that have less execution time than the code version with the highest reliability. Finally, we sort the selected code versions in order that they provide the maximum execution time reduction with minimum reliability loss comparing to the version with maximum reliability (in descending order of time/reliability).

Algorithm Discussion: Algorithm 1 shows the pseudo-code of the run-time phase of our dsReliM system. At the initialization step, we map tasks from minimum reliability to maximum reliability (sorted in line 1) on cores from highest performance to lowest performance (sorted in line 2) to achieve high reliability with best effort. Also, in the initial mapping, the code versions with the maximum reliability are considered and the maximum V-f levels are assigned to the cores (line 4). In each iteration of the mapping, the code version with the lowest performance is mapped to a core (line 5). A similar mapping is performed for the highest reliability code version (line 6). Then, we consider the remaining tasks in the order of decreasing reliabilities (line 7). In each iteration, we map tasks from minimum to maximum reliabilities (line 8). Finally, tasks with a deadline that is not met are mapped to cores with maximum reliabilities (line 9). After mapping all tasks, we update the code version and V-f levels (line 10). The algorithm continues until all tasks are mapped (line 11).
main for loop, the index \( \{q\cdots q+n-1\} \) for \( X \), represents a window of \( n \) cores that are considered for mapping \( n \) ready tasks. In the first iteration (\( q=1 \)), the first \( n \) performance-wise best cores are considered. In each iteration, if the TDP constraint is not met with the underlying cores, the index \( q \) increases to drop the core with the maximum power consumption and add a core with lower power to the window. After initialization, we find appropriate V-f level scalings and their associated code changes (lines 5-20). Each V-f scaling is associated with a priority of \( \text{Apower/\text{A}reliability} \) to determine the maximum power reduction with minimum reliability loss (line 9). Also, if the V-f level scaling leads to a deadline miss, a code version change is involved in the V-f level scaling (lines 11-17). After determining all V-f level scalings and code changes for all ready tasks and then putting them into a max-heap. Therefore, for \( n \) ready tasks and 1 V-f levels, building the max-heap is performed in \( O(n \times l) \). Note that, since the code versions are sorted at design-time, finding an appropriate code version through lines 11-17 of Algorithm 1 can be done in \( O(1) \). The main for loop (line 3) iterates for \( c \times n \times l \) times. In case that the number of free cores \( c \) is far more than the number ready tasks \( n \) (\( c \gg n \)), the main loop is done in \( O(c) \). Note that, in case of high loads, e.g., when the number of ready tasks is equal to the number of free cores (\( n=c \)), the main for loop executes only once. Therefore, the number of iterations for the main loop highly depends upon the system load. Building the max-heaps for all possible mappings can be done in \( O(c \times n \times l) \). Finally, the algorithm removes the V-f level scalings and code changes from the heap one after another and checks whether the TDP constraint is met. This step is also done in \( O(c \times n \times l) \). Therefore, the order of the algorithm is \( \max \{O(c \log c), O(n \log n), O(c \times n \times l)\} \). Since in this paper it is assumed that the algorithm is performed when the number of free cores is greater than the number ready tasks (i.e. \( c \geq n \)), we have \( c \geq \log(n) \) and hence: \( c \times n \geq n \log(n) \). Therefore, the total order of the algorithm is \( \max \{O(c \log c), O(c \times n \times l)\} \).

### 4. EXPERIMENTAL RESULTS

#### 4.1 Experimental Setup

We conducted experiments on various process variation maps to evaluate our dsReliM system using a system-level many-core simulator. In order to provide accurate power and performance characterization for the simulator, we did ASIC synthesis and gate-level simulations. To achieve power values at different V-f levels we synthesized a VHDL implementation of LEON3 processor using the Synopsis Design Compiler and the TSMC 45nm low-power standard cell library. We adopted a power model based on Eq. 4 and the power values of Table 1. Also, to acquire applications activity factors we used ModelSim for gate-level simulation. In our experiments transient fault rate at different V-f levels was modeled based on Eq. 1 with \( \lambda = 10^{-4} \) and \( \Delta = 1 \) [19].

To generate process variation maps the model in the work [29] was used considering 80\times80 grid cells chips with correlation parameter 0.5 and standard deviation of process variation 10% of the nominal process parameter [30]. To model variations on the

![Accuracy comparison (low workload)](image)

![Accuracy comparison (high workload)](image)

![Runtime efficiency comparison (low workload)](image)

![Runtime efficiency comparison (high workload)](image)

Figure 7: Accuracy and runtime efficiency comparison with ILP solver.

<table>
<thead>
<tr>
<th>(V-f) Level</th>
<th>Power Consumption [mW]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dynamic</td>
</tr>
<tr>
<td>[0.72, 490]</td>
<td>140.62</td>
</tr>
<tr>
<td>[0.85, 650]</td>
<td>228.78</td>
</tr>
<tr>
<td>[0.97, 730]</td>
<td>336.81</td>
</tr>
<tr>
<td>[1.10, 850]</td>
<td>463.17</td>
</tr>
<tr>
<td>[1.23, 970]</td>
<td>641.39</td>
</tr>
</tbody>
</table>

Table 1. Power consumption of Leon3 processor core.
maximum frequencies and leakage power due to process variations we did SPICE simulations. First, we implemented a 13-stage ring oscillator containing FO4 inverters based on 2-input N-AND gates (like the works [31][32]) using the 45nm metal gate/high-k Predictive Technology Model (PTM) [33]. Then, we did curve fitting on the simulations points to achieve leakage power and frequency values [34]. We considered that the frequencies vary from 400MHz to 1GHz for 0.72V and 1.23V (the lowest and highest levels), with 10MHz frequency binning step. Figure 1a shows an example for generated leakage power variations for a 10×10 core processor. Also, the frequency variations of the first 16 cores are shown in Figure 1b. To consider a wide range of different execution conditions, we used various combinations of real-world and synthetic applications. The real-world applications are: ADPCM, SHA, SUSAN, CRC, SATD, and SATD. The synthetic task graphs were generated using TGFF [35]. In the experiments, it is assumed that the software vulnerability for the synthetic tasks and different code versions of the tasks varies between 0.06 and 0.2 and the clock cycles vary between 2K and 2M, i.e. the minimum and maximum values of that for the real-work applications.

4.2 Comparison with ILP solver

As discussed in Section 3.2, since the runtime complexity of ILP solvers may increase exponentially with the increase of problem, they cannot be employed in online scenarios. Therefore, like the work in [28], to evaluate the accuracy and runtime efficiency of our dsReliM system in finding solutions at run time that are close to optimum, we compared it with a well-known ILP solver (Gurobi [25]). In this experiment, the power consumption when all cores perform at their maximum V-f level was considered as the baseline chip power. The TDP budget is considered to vary between 40% and 100% of the baseline chip power to represent the range from high TDP constraints (60% Dark Silicon) to no TDP constraint (all cores can perform on their maximum available frequency, 0% Dark Silicon). Also, to achieve a baseline for system reliability, 500 random mappings (random tasks, code versions, task-to-core assignments and V-f levels) are generated and the average reliability of the 500 random mappings is considered. The reliability values for our dsReliM system and the ILP solver are normalized to the baseline reliability in Figure 7. To evaluate the runtime efficiency of our dsReliM system, the execution time of both our system and the ILP solver when running on a dual-core PC (2.6GHz, 2GB memory) running 64-bit Ubuntu Linux is reported in Figure 7. In this experiment we considered two types of system workload: 1) low workload, when the number of ready tasks is less than 70% of the number of free cores and 2) high workload, when the number of ready tasks is more than 70% of free cores. From the accuracy point of view, on average, dsReliM provides solutions that differ about 2.5% and 0.7% from optimum for low and high workloads, respectively. Also, dsReliM achieves up to 12% and 18% reliability improvement compared with the baseline for low and high workloads. From runtime efficiency point of view, on average, dsReliM provides 1500 and 3100 times faster than the ILP solver for low and high workloads. As it can be concluded form Figure 7, dsReliM achieves more efficiency in high workloads. This is because, under a high workload the main loop of Algorithm 1 iterates for a fixed number of times to find the solution.

4.3 Comparison with State-of-the-Art

We also compared our proposed dsReliM system with a state-of-the-art system that is presented in [8] from both reliability improvement and runtime efficiency points of view. The system of [8] implements a heuristic-base algorithm to integrate application mapping with DVS to optimize system performance and power, while satisfying chip TDP constraints and application-specific reliability constraints. In this experiment the TDP budget is considered to vary between 40% and 100% of the baseline power consumption (i.e. the case when all cores perform on their maximum frequency) and the results are shown in Figure 8. In Figure 8.1 the reliability values for the systems are normalized to the baseline reliability, i.e. the average reliability for 500 random code version and V-f level assignment and task-to-core mapping. Experiment results show that, on average, dsReliM provides 19% reliability improvement compared to the system of [8] (Figure 8a) with only 8% decrease in runtime efficiency (Figure 8b).

5. CONCLUSION

This paper presents a reliability management system for Dark Silicon chips (dsReliM) that accounts for soft errors, process variations and the thermal design power (TDP) constraint. It leverages multiple reliable application versions executing at different V-f levels as power-reliability optimization knob. We modeled the reliability management problem as a bounded 0-1 integer linear program (ILP) and further proposed a computationally lightweight yet efficient heuristic-based algorithm for solving the problem. Results have shown that compared to an optimal solution, dsReliM deviates up to 2.5% in terms of reliability efficiency, but speeds up the reliability management decision time by a factor of up to 3100. dsReliM also provides up to 19% reliability improvement under different TDP constraints when compared to a state-of-the-art technique.

6. Acknowledgment

This work is supported in parts by the German Research Foundation (DFG) as part of the priority program “Dependable Embedded Systems” (SPP 1500 - spp1500.itec.kit.edu).

7. REFERENCES


