ASER: Adaptive Soft Error Resilience for Reliability-Heterogeneous Processes in the Dark Silicon Era

Florian Kriebel, Semeen Rehman, Duo Sun, Muhammad Shafique, Jörg Henkel
Chair for Embedded Systems (CES), Karlsruhe Institute of Technology (KIT), Germany
{florian.kriebel, muhammad.shafique, henkel}@kit.edu; {semeen.rehman, duo.sun} @student.kit.edu

Abstract—The Dark Silicon provides opportunities to realize Reliability-Heterogeneous Processors with ISA compatible cores having different levels of protection against reliability threats (like soft errors). This paper presents design-time customization of Reliability-Heterogeneous Processors given a set of applications and area constraints. A run-time system adaptively manages the soft error resilience under a given thermal design power (TDP) budget. We synthesize an embedded processor with different levels of protection and present area and power results for a 45nm technology. We illustrate the benefits of adaptive soft error resilience by comparing with four different state-of-the-art approaches where we achieve 58%-96% overall system reliability improvements under a tight TDP constraint (corresponding to a 65% dark area).

I. INTRODUCTION AND RELATED WORK

In the nano-era, reliability has become one of the primary design concerns for on-chip systems that are subjected to several reliability issues like soft errors, aging, etc. [1]-[3]. With technology scaling, the total soft error rate for a chip is envisaged to increase due to shrinking transistor dimensions and lower voltages that lead to smaller critical charges [1]. Soft errors are transient faults in the hardware due to high energy particle strikes and typically manifest as bit flips. These faults propagate towards the software layers and corrupt the application program’s execution. In order to mitigate the soft errors, plenty of work has been performed at both software and hardware levels [5]-[8]. These techniques primarily rely on redundancy/checking, thus incur significant performance and/or area/power overhead. For manycore on-chip systems, soft error resilience is realized through hardened/reliable cores [8][9] and exploiting idle cores for spatial redundancy [10]-[13]. To effectively use the idle cores in a manycore system, techniques like Redundant Multithreading (RMT) [12][13], process-level redundancy [10], and adaptive core redundancy [11] have been proposed. The RMT can be realized in two different ways: (i) temporal redundancy using SRT (simultaneous redundant multithreading) on the same core [12]; and (ii) spatial redundancy using CRT (chip-level redundant multithreading) [13]. According to [13], CRT is preferred over SRT due to its lower performance penalty. The above-discussed techniques assume that excessive cores are available for every application thread for either redundancy using CRT [13] or protection using reliable cores [8]. However, this may not be possible under scenarios of massive multi-threading (i.e., 10s-100s of threads) and limited power budgets. As a result, in such scenarios, not all applications may be supported with full DMR/TMR (dual/triple modular redundancy) or CRT. Moreover, these techniques either provide the same level of reliability for all applications or no reliability due to their inherent limitation of supporting only a particular reliability mode.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

DAC ’14, June 01 - 05 2014, San Francisco, CA, USA
Copyright 2014 ACM 978-1-4503-2730-5/14/06...$15.00.
http://dx.doi.org/10.1145/2593069.2593094

Motivational Analysis: Our program reliability analysis in Fig.1 illustrates that different applications exhibit diverse vulnerabilities (i.e., probability of an error during the execution of an application program) and error masking properties1. Fig.1 shows that the “Susan” application has higher but varying vulnerabilities compared to the “ADPCM” application due to its distinct data and control flow properties. However, the error masking probabilities are also high for “Susan” due to specific instruction profile (with logical instructions that mask most of the errors in the preceding instructions in the data flow). As a result, “Susan” application encounters only 5% output errors while “ADPCM” encounters 45% output errors in our Monte-Carlo based fault injection experiments even at a very high fault rate of 5faults/MCycle (see detailed experimental setup in Section VI).

In summary, our reliability analysis shows that different applications have different error occurrence and masking probabilities and not all applications require the same level of reliability from the underlying hardware. Therefore, in order to address the limitations of above-discussed techniques, it may be beneficial to have a set of reliability-customized cores that provide distinct reliability, performance, power, and area properties and can be dynamically allocated to different applications based on their vulnerability and masking properties. However, provisioning cores with various reliability/protection levels may incur significant silicon area cost, for which a solution can be found in the Dark Silicon chips [4].

Dark Silicon chips refer to chips with an abundance of transistors due to aggressive technology scaling but only a fraction of that can be simultaneously powered-ON under a given Thermal Design Power (TDP) constraint [15][16]. Based on the technological data from ITRS and Intel, authors in [15] have done analytical predictions of dark silicon according to which 1nm-8nm chips will have >50%-80% dark silicon. Recent trends predict an early evidence and feasibility of leveraging the dark silicon for architectural heterogeneity, and specialized cores with a focus on performance-power-efficient design [17][21]. Prominent approaches include: accelerator rich architectures [17], TDP budgeting for performance improvement [19], exploitation of process variations, and architectural/device heterogeneity for performance and power [18][20][21][23]. However, to the best of our knowledge, state-of-the-art has not yet explored the opportunities offered by dark silicon for

1 The vulnerabilities and masking probabilities are computed using the program reliability models provided in [6][14].
2 The thermal design power (TDP) refers to the total power provided to a chip that can be effectively dissipated with a given cooling setup.
customizing general-purpose manycore processors towards reliability-heterogeneity and run-time soft error resilience under TDP constraints. This paper makes the first attempt to bridge this gap and address the above-discussed scientific challenges.

**Associated Challenges for Soft-Error Resilience in Dark Silicon Chips:** Leveraging dark silicon for adaptive soft error resilience introduces the following two key challenges:

1) **At Design-Time:** For efficient use of the dark silicon area to provide reliability means, there is a need for architectural customization towards integrating many iso-ISA\(^3\) general-purpose cores but with heterogeneous reliability features (e.g., architectural support for DMR/TMR, pipeline protection, CTR, etc.). The customization and architectural template synthesis problem needs to account for (i) given dark silicon area and TDP constraints; (ii) available library of cores with different reliability features; and (iii) a set of representative benchmark applications with diverse soft error vulnerability, error masking, performance properties.

2) **At Run-Time:** Given such reliability-customized architecture and run-time TDP budget, another challenge is to efficiently allocate appropriate core types to different concurrently executing applications while accounting for their varying reliability and performance properties. It may be beneficial to allocate high-reliability cores to applications\(^2\) with low masking and high vulnerability properties, while allocating low-reliability core to applications with high masking and low vulnerability properties, such that the overall system’s reliability is maximized under the given TDP budget. Note that such an allocation decision cannot be taken at design time due to the following run-time uncertainties: (i) changing workloads and performance requirements of applications; (ii) their varying vulnerabilities and masking properties due to changing data and control flow; (iii) varying set of concurrently executing applications; (iv) available power budgets; and (v) raw fault rate that depends upon location and operational conditions of the device.

**Our Novel Contributions and Concept Overview**

We address the above-discussed challenges with our novel Adaptive Soft Error Resilience (ASER) approach for Reliability-Heterogeneous Dark Silicon Manycore Processors (darkRHPs). ASER realizes efficient reliability optimization in the dark silicon era through design-time and run-time optimization by means of the following two components: (1) design-time customization of darkRHPs; and (2) TDP-constrained adaptation of soft error resilience at run time considering distinct application vulnerabilities.

**Our novel contributions in a nutshell are:**

1) A design-time methodology (Section IV) for customization of reliability-heterogeneous dark silicon manycore processor (darkRHPs) that, for a given library of cores, selects a set of cores with heterogeneous area, power and reliability properties while maximizing the total reliability of a set of representative benchmark applications under available area and TDP constraints. We formally model the problem of customization as a Bounded Knapsack Problem and employ a heuristic to generate various templates under a given set of constraints. The goal is to jointly maximize the reliability (both functional and timing) of the given set of benchmark applications. To address run-time uncertainties, the darkRHPs employ:

2) An adaptive soft error resiliency manager (Section V) that, for an execution scenario at run time, allocates a set of cores with particular reliability modes to different concurrently executing applications depending upon their varying performance requirements and vulnerability properties under a given TDP constraint.

To evaluate our ASER approach, we have developed a library containing various versions of an embedded processor with different reliability levels in VHDL (see Section VI.A) and synthesized using an ASIC design flow. This library serves as an input to our design-time customization methodology to generate different templates of darkRHPs (Section VI.B), which are then evaluated for run-time adaptations under different TDP constraints (Section VI.C). To the best of our knowledge, ASER is the first work towards customization and run-time management of reliability-heterogeneous dark silicon manycore processors.

**II. SYSTEM MODELS AND PRELIMINARIES**

**Hardware Architecture and Fault Models:** Our architectural model assumes a darkRHP processor with N iso-ISA RISC-like embedded processor cores (e.g., LEON3) with private instruction and data cache. The set of cores is given as \( \mathcal{C} = \{C_2, C_3, \ldots, C_N\} \), such that \( C_i \) \( \forall i \in \{1, K\} \), denotes a core type that is heterogeneous w.r.t. its reliability functionality which is achieved by protecting\(^2\) different parts of the processors (e.g., full pipeline protection, full register file protection, full cache protection, or a combination of any). \( N(C_i) \) denotes the number of available cores of type \( C_i \) and the total number of cores \( N \) in a darkRHP is given as \( N = \sum_{i = 1}^{4} N(C_i) \).

The area and frequency of each core type \( C_i \) are given as \( A(C_i) \) and \( F(C_i) \), respectively. Note that a core may also be heterogeneous w.r.t. its synthesizable frequency and/or process corner settings, such that a core has different area and power features compared to its baseline, i.e., unprotected core. For the generalization purpose, we consider these cores also as different variants and leave it to the library building step (which is not the focus of this paper, we assume a given library, though for the experimental setup, we develop our own). A darkRHP processor is envisaged to have a plethora of cores but only a subset \( C \) can be powered-ON at run-time under a given Thermal Design Power (TDP) constraint.

**Uncore Part:** In this paper, our focus is on optimizing the number and type of cores (including their private caches). Remaining memory hierarchy is assumed as shared and kept constant, thus placement of different tasks has less impact on the overall performance \[21\]. Therefore, without loss of generality, we assume a fixed area and peak power budgets for the uncore part which acts as a fixed cost. Therefore, the optimization is done for the TDP and area budgets of the cores.

**Fault Model:** We consider transient faults/soft errors, single- and multiple-bit upsets in both the combinatorial and sequential logic.

**Application Model:** We consider a set of task graphs \( \{T, E\} \) containing tasks and dependency information for all given applications. \( T \) is a set of \( M \) tasks, such that \( T = \{T_1, T_2, \ldots, T_M\} \). \( E \) is the set of task dependencies: \( E = \{E_{ij} \mid (T_i, T_j) \in \mathcal{T}) \). For a given core type \( C_i \), each task \( T_j \) has following properties: \( P(T_j, C_i) \) is the peak power consumption, \( L(T_j, C_i) \) is the average case performance in terms of execution time, and \( TRF(T_j, C_i) \) is the task reliability factor considering the protection level provided by the core type \( C_i \) each application task executes on its dedicated core.

**Application Reliability Model:** In order to quantify the reliability of an application task \( T_j \) executing on a core type \( C_i \), we employ the task reliability factor \( TRF(T_j, C_i) \), which is based on the software reliability estimation model of\([7]\); see Eq. 1. It jointly accounts for the functional reliability (i.e., task vulnerability to soft errors) and timing reliability (i.e., probability of deadline misses, \( P_{D0M}(T_j, C_i)\)). \( \alpha \) is a user-defined parameter to prioritize functional or timing reliability. Note that a smaller \( TRF \) value corresponds to a more reliable task execution.

---

\(^2\) Iso-ISA cores refer to the cores that have same instruction set architecture.

\(^3\) A comprehensive survey of application mapping policies can be found in \[25\].
$TRF(T_j,C_i) = \alpha \times FVI(T_j,C_i) + (1-\alpha) \times P_{RUN}(T_j,C_i)$ \hspace{1cm} (1)

The $TRF$ model of [7] quantifies the task vulnerability using a Function Vulnerability Index [6], i.e., $FVI(T_j,C_i)$, that denotes the probability of soft errors during the task execution (Eq. 2) considering the vulnerability of each instruction ($W_i$, Eq. 2, 3) executing on different components of the processors that may be protected or unprotected. In case, a processor component is protected, the instruction vulnerability within this component is considered as “0”. Otherwise, it depends upon the spatial vulnerability (i.e., area-wise error probability considering the knowledge of vulnerable bits) and temporal vulnerability (i.e., time-wise error probability considering the vulnerable time period an instruction spends in a pipeline component).

$FVI(T_j,C_i) = \frac{\sum_{\text{inst}\in T_j} \left(WVI_{\text{inst}}(T_j,C_i)\right)}{N_{\text{inst}}(T_j)}$ \hspace{1cm} (2)

$WVI_{\text{inst}}(T_j,C_i) = \sum_{\forall \text{core}\in C_i} (WVI_{\text{inst},c}\times A_c) / \sum_{\forall \text{core}\in C_i} A_c$ \hspace{1cm} (3)

$WVI_{\text{inst}}(T_j,C_i)$ is the instruction vulnerability of an instruction $\text{inst}$ of Task $T_j$ executing on core type $C_i$. $\rho$ is the error occurrence probability of the component (considering gate-level logical masking) which is given as “0” in case the component is protected, and $A_i$ is the logic area of the core. More details on these models can be found in [6][7].

III. ASER SYSTEM OVERVIEW

We leverage the given dark silicon area to design and manage so-called reliability-heterogeneous dark silicon manycore processor (darkRHPs) using multiple iso-ISA but reliability-heterogeneous cores. Fig.2 illustrates the overall flow of our methodology showing different design-time and run-time steps.

**Design-Time Steps:** First a library of different iso-ISA cores with different reliability features is generated. A core can be fully hardened (using TMR), or partly hardened (i.e., pipeline protection, cache protection, register file protection, heterogeneous/selective rollback recovery [22] or a combination of these) given different area constraints, or unhardened (i.e., the baseline case). Without the loss of generality, we employ TMR to different parts of processor for generating different partly hardened core types. Fig.3(a) shows a few example reliability-heterogeneous LEON3 cores along with their area and power consumption (more details on processor synthesis are provided in Section VI.A).

**Run-Time Steps:** Although a darkRHP exhibits many cores of different types, not all may be powered-ON simultaneously for a given set of concurrently executing applications under a TDP constraint. Therefore, our darkRHP employs run-time Adaptive Soft Error Resilience (ASER) manager that, under a given TDP budget, selects a subset of cores of different types depending upon the requirements of the concurrently executing applications to improve the overall system reliability (i.e., total $TRF$ is minimized) and remaining cores are kept dark. It accounts for the $TRF$ of different applications during the allocation process. For instance, a fully or partly hardened core may be allocated to an application task with high $TRF$ value and vice versa. Fig.4 illustrates some abstract example scenarios for different run-time contexts in a darkRHP.

In the following, we discuss our darkRHP customization and adaptive soft error resiliency manager in details.

IV. CUSTOMIZATION OF DARKRHPs

We formulate the problem of customization/architectural synthesis of darkRHPs as the Bounded Knapsack Problem (BKP). The knapsack is the darkRHP chip with total area capacity $A_{\text{const}}$. The objects in the knapsack are different types of cores, $C\in[C_1, ..., C_d]$, such that, each type of core $C_i$ has multiple copies.

**Bounds or Number of Copies for Different Object Types:** In BKP each object is associated with a bound $N_{\text{ub}}(C_i)$, which is a non-negative number. In our case, $N_{\text{ub}}(C_i)$ is given as the maximum number of a core type that can fit within the given $A_{\text{const}}$.

$\forall C_i \in C, N_{\text{ub}}(C_i) = A_{\text{const}} / A_{C_i}$ \hspace{1cm} (4)

**Capacity / Area Constraint:** The total area, i.e., the sum of all selected cores in the candidate solution $C'=[C_1', ..., C_d']$ should be less than or equal to $A_{\text{const}}$.

$\sum_{\forall C_i \in C} (N(c)\times A(c)) \leq A_{\text{const}}; \text{ s.t. } N(c) \leq N_{\text{ub}}(c)$ \hspace{1cm} (5)

**Optimization Goal:** The set of cores $C'$ that minimizes the total $TRF$ for all applications is selected, i.e.,

$\min \sum_{\forall C_i \in C} \left(\sum_{c=1}^{N(C)} \left(\sum_{T_j \in T} TRF(T_j,c,x)\right)\right)$ \hspace{1cm} (6)

**Power-Wise Feasible Solution:** At the design time, a darkRHP chip can have more cores due to available silicon area. However not all cores can be powered-ON at the same time during the run time due to the TDP constraint. Therefore, the $TDP$ constraint defines the run-time chip context; while $A_{\text{const}}$ defines the design-time chip context. In case the system designer expects a minimum set of applications $M'$ to be always available, we can enhance the above problem with a so-called “TDP-feasibility constraint”. It ensures that at any given point
in time where $M'$ applications are executing concurrently; the selected core type $C'$ can be used by any other application (i.e., $T - M'$) without violating the TDP constraint. Otherwise it may not be beneficial to select such a core type. This "TDP-feasibility constraint" can be formulated as Eq. 7, where for a candidate core $C'$, at any given point in time, the total peak power consumption for all given applications must be less than or equal to TDP. 

$$\forall T_j \in \{T - M', \ P(T_j, C') + \sum_{T_l \in M'} P(T_l, C_l)) \leq TDP$$  \hspace{1cm} (7)

Customization Algorithm: Optimally solving the BKP problem is NP-hard, therefore, we develop a heuristic algorithm (see Algorithm 1). The inputs are: (i) a set of ISO-ISA core types $C = \{C_2, ..., C_k\};$ (ii) a set of $M$ representative benchmark applications $T = \{T_2, T_2, ..., T_M\}$ along with their peak power, average case performance, and reliability properties; (iii) total available area $A_{const}$; and (iv) the thermal design power $TDP$. The outcome of the customization step is the darkRHP architectural template with the set of selected cores $C'$. Before starting the algorithm the following initial conditions are checked and the algorithm executes only if necessary.

Initial Condition – 1: If there is sufficient area available to support all core types for all applications then simply allocate it and return. A more feasible scenario to execute Algorithm 1 is:

$$\sum_{C_i \in C}(A(C_i) \times M) >> A_{const}$$.  \hspace{1cm} (8)

Initial Condition – 2: If there is only area for one baseline core $C_i$ per application then simply allocate and return. In the dark silicon chips, sufficient area is available to provide many instances/copies of different core types for different applications.

$$A(C_i) \times M << A_{const}$$.  \hspace{1cm} (9)

Initial Condition – 3: Allocate one baseline core $C_i$ for each application to ensure that minimal operating requirements of applications are met under the TDP capacity.

$$C' = \{C_1, ..., C_k\}, \ s.t., N(C_i) = M \ and \ \forall C_i \in \{C - C_i\} \ N(C_i) = 0;$$  \hspace{1cm} (10)

Initial Condition – 4: The total peak power consumption for all applications executing on the baseline cores $C_i$ should be significantly less than the TDP constraint. Only then it is beneficial/feasible to select a core type with a certain level of reliability/protection.

$$\sum_{T_j \in C}(P(T_j, C_i)) << TDP$$.  \hspace{1cm} (11)

**Algorithm 1: Customization Algorithm**

**INPUT:** Set of cores $C = \{C_1, ..., C_k\}$, set of application tasks $T = \{T_1, T_2, ..., T_M\}$, available area constraint $A_{const}$ and thermal design power $TDP$

**OUTPUT:** $C' = \{C_1, ..., C_k\}$ - darkRHP arch. template with selected cores.

**BEGIN**

1. for all $C_i \in C$ do //initialize TRF profit values for all core types
2. \hspace{1cm} profit($C_i$) = $\sum_{T \in C}(T_{max}TRF(T_j, C_i))$; end for
3. for all $C_i \in C$ do $C_i \leftarrow 0$; end for
4. $N(C_i) \leftarrow M$; $N(C_i) - M = A - M + A(C_i)$; // initialization
5. $P \leftarrow \sum_{T \in C}(T_{max}TRF(T_i, C_i))$; $x \leftarrow 0$; $A' \leftarrow 0$;
6. $C' \leftarrow \text{sort}(C \text{profit, descendingOrder})$;
7. while $(x \leq N) \&(A' \leq A_{const})$ do //loop until the area is exhausted
8. \hspace{1cm} $k \leftarrow C_i$; //Get the core type of a current best profit
9. \hspace{1cm} $A' \leftarrow A' + A(C_i)$;
10. \hspace{1cm} if $(A' \leq A_{const}) \& (\forall T_j \in \{T - T'\} \ P(T_j, C_i) + p \leq TDP)$ then
11. \hspace{1cm} $A \leftarrow A + A(C_i)$;
12. \hspace{1cm} $N(C_i) = +$;
13. \hspace{1cm} $N(C_i) = -$;
14. else $x++$; end if
15. end while
16. END

**Algorithm 2: Adaptive Soft Error Resilience Algorithm**

**INPUT:** Architectural template with different types of cores $C = \{C_1, ..., C_k\}$, Set of run-time concurrently executing application tasks $T = \{T_1, ..., T_M\}$, available thermal design power $TDP$

**OUTPUT:** $C' = \{C_1, ..., C_k\}$ - darkRHP arch. run-time chip context.

**BEGIN**

1. for all $t \in T$ do //initialize core assignment
2. \hspace{1cm} $x(t) \leftarrow C_i$; estimate TRF$(t, C_i)$ and $P(t, C_i)$; end for
3. for all $t \in T$ do //select the best core under the power cap
4. \hspace{1cm} $P_R \leftarrow TDP - \sum_{T \in C}(T_{max}TRF(t, C_i))$;
5. \hspace{1cm} $T' \leftarrow \text{sort}(T, \text{TRF, ascendingOrder})$;
6. for all $t \in T'$ do //Find an appropriate core for each task
7. \hspace{1cm} TRFBest $\leftarrow TRF(t, x(t))$;
8. for all $C_i \in C'$ do //Select the best core under the power cap
9. \hspace{1cm} if $(P_R - P(t, C_i) + P(t, C_i) \geq 0) \&(N(C_i) > 0) \&(TRF(t, C_i) < TRFBest)$ then
10. \hspace{1cm} $C_{Best} \leftarrow C_i$;
11. \hspace{1cm} TRFBest $\leftarrow TRF(t, C_i) < TRFBest$;
12. end if
13. end for
14. if $(C_{Best} \neq C_i)$ then
15. \hspace{1cm} $x(t) \leftarrow C_{Best};$
16. end if
17. end for
18. END

a core type with a certain level of reliability/protection. The average case time complexity of our algorithm is $O(N.m.m) = \text{arg}(N_{TR}(C_i))$, while the best case complexity is $O(N.m)$.

V. ASER: ADAPTIVE SOFT ERROR RESILIENCY MANAGER

After the template of a darkRHP is obtained, our run-time soft error resiliency manager (Algorithm 2) allocates a particular type of core to each application task such that the total system reliability is maximized (i.e., minimizing the total TRF while meeting tasks’ deadlines) under the given TDP constraint (i.e., the sum of peak powers of all applications is less than or equal to TDP); see line 9). Despite of the application specific customization in Section IV, the decision about which application gets which type of core cannot be determined at design-time” due to the changing workloads, vulnerabilities, control flow variations, set of active applications, available power budgets, and raw error rates. The input to our run-time manager
is the template of the darkRHP obtained from Section IV, the set of concurrently executing/active applications (different from the set of representative application benchmark used for design-time customization), and the TDP constraint. The core allocation function is given as:

\[ X: T^* \rightarrow C^*; \quad \forall t \in T^*, X(t) = C^*_t; \quad \text{s.t.} \quad C^*_t \in C^* \]

under the constraint that only one application task executes on one core.

The core assignment loop executes for all tasks sorted w.r.t. their TRF, i.e., in a priority order where the tasks with the lowest reliability obtains the highest priority (lines 5-18). For each task, there may exist multiple solutions. The goal is to find an allocation that minimizes the total TRF while keeping the total peak power below the TDP capacity (lines 8-13)

\[ \text{minimize} \left( \sum_{t \in T^*} \text{TRF}(t, X(t)) \right) \]

(12)

VI. RESULTS AND DISCUSSION

A. Processor Synthesis, Library Building, and darkRHPs

Reliability Simulation and Evaluation Setup

Fig. 5 demonstrates our integrated tool flow and simulation setup with processor synthesis, darkRHP customization and run-time darkRHP reliability simulations.

Processor Versions and Synthesis: We have extended the online available LEON3 code by Gaisler research labs for eight different reliability versions. Note that this effort has been done manually, therefore, a wide set of reliability features is not possible due to lack of time. These eight VHDL projects were individually synthesized using Synopsys Design Compiler and TSMC 45nm technology library for four different frequency settings and three available process corners. The area and (leakage, dynamic, and total) power results for process corner 3 (settings are: 0.81V, 125 °C junction temperature, Corner SS) are shown in Table I. The configuration of the baseline processor is: core-private instruction and data caches with configuration=1set, 4Kbyte/set and 32Byte/line. The MMU configurations are: TLB entries for 8 instruction and 8 data, fast write buffer and 4KMMU page size.

darkRHP Reliability Simulation and Evaluation Setup: Our darkRHP simulator enhances the ArchC based LEON3 ISS simulator with fault generation and injection modules (see details in [6]). We perform Monte-Carlo fault injection experiments with different fault rates depending upon the device location and altitude. Using the flux calculator [5], we obtain the neutron flux rate and for a given frequency and chip area, and fault models (single or multiple bit flips), etc. we obtain the number of fault for each simulations. We have performed 1000s of different fault injection experiments per processor component. We have deployed various applications from an embedded MiBench Benchmark like “ADPCM”, “SUSAN”, “AES”, “SHA”, “CRC”, and a complex video encoder “H.264”. We have several mixes of these applications to realize random and real-world scenarios like secure video processing, such that each application has several threads executing in parallel.
compare it with four state-of-the-art run-time reliability optimizing techniques for our given template. The first one is a timing reliability-optimizing (TRO) technique that aims at minimizing the probability of deadline misses (with EDF scheduling) [7]. Second is an approach that optimizes for functional and timing reliability. We name it as RTO [7]. Third is full TMR [13] and fourth is adaptive TMR (adTMR) that deactivates the TMR functionality given the vulnerability is below a specified threshold [8]. We use the metric Reliability Profit Function (RPF) as defined below, higher is better. 

\[
RPF = 1 - \left( \frac{RTP(TRO)}{RTP(adTMR)} \right) \cdot \varepsilon \cdot \epsilon \cdot Z \in \{ \text{TRO, RTO, TMR, adTMR} \}
\]

VI. CONCLUSIONS

We presented a novel ASER approach for design-time customization of reliability-heterogeneous dark silicon manycore processors (darkRHPs) and their run-time management for adaptive soft error resilience under dark silicon area and TDP constraints. The key is to leverage the variable reliability and performance properties of different applications. Compared to four different state-of-the-art techniques, our ASER achieves 58%-96% improvements in the system reliability. We demonstrate that dark silicon provides opportunities for developing highly dependable systems with reliability-heterogeneous processors. The key is to leverage the application specific reliability properties to achieve high system reliability under a given TDP constraint at run time.

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).

REFERENCES