8th International Conference on Smart Cities, Internet of Things and Applications (SCIoT2024)
Department of Engineering, Ferdowsi University of Mashhad (FUM), Mashhad, IRAN.
May 14-15, 2024

Reliability analysis in ARM big.LITTLE heterogeneous multi-core processor for IoT

Mahin Moradiyan
Dependable Distributed Embedded Systems (DDEmS) Laboratory
Computer Engineering Department
Ferdowsi University of Mashhad
Mashhad, Iran
m_moradiyan@mail.um.ac.ir

Yaser Sedaghat
Dependable Distributed Embedded Systems (DDEmS) Laboratory
Computer Engineering Department
Ferdowsi University of Mashhad
Mashhad, Iran
y_sedaghat@um.ac.ir

2024 8th International Conference on Smart Cities, Internet of Things and Applications (SCIoT) | 979-8-3503-7467-4/24/$31.00 ©2024 IEEE | DOI: 10.1109/SCIoT62588.2024.10570143

***Abstract*— This study navigates the landscape of the Internet of Things (IoT), spotlighting the escalating demand for robust processing solutions, thereby propelling the pervasive adoption of heterogeneous multi-core processors. Within this sphere, the ARM big.LITTLE architecture takes center stage, adeptly harmonizing high performance with energy efficiency. The investigation zooms in on the pivotal role of failure-sensitive applications in the IoT realm, critical domains like healthcare. The paper underscores the indispensable need for failure analysis concerning ARM big.LITTLE processors, unravel the complexities of data flow errors (DFE) and control flow errors (CFE). By providing insights into potential pitfalls and strategies for fortifying reliability, the study contributes to the evolving discourse on the resilience of ARM big.LITTLE processors in the context of failure-sensitive IoT applications. Also concludes by emphasizing the need for continued exploration of optimal task mapping and fault tolerance strategies, paving the way for future research to refine and extend the reliability analysis of ARM big.LITTLE platforms in evolving IoT landscapes.**

Keywords—IoT, Reliability, Heterogenous Multi-core processor, ARM big.LITTLE, ARM Cortex-A15, ARM Cortex A-7.

I. INTRODUCTION

The relentless evolution of technology, underscored by the emergence of the Internet of Things (IoT) and the advent of smart cities, has propelled us into an era where interconnectedness and data-driven capabilities define our digital landscape [1], [2]. In navigating this transformative journey, the reliability of computing systems takes center stage, particularly in the intricate tapestry of IoT and smart city environments. As the volume of data, diverse workloads, and the integration of intelligent devices surge unprecedentedly, the demand for robust processing architectures becomes paramount. Heterogeneous multi-core processors, a linchpin in addressing the computational demands of this connected world, promise a delicate balance between performance and efficiency [3], [4].

Departing from conventional monolithic architectures, heterogeneous multi-core processors usher in a paradigm shift in computational capabilities. Characterized by a diversity of cores optimized for distinct tasks, these processors enable the concurrent execution of a myriad of applications. In the context of IoT, where a multitude of devices collaborate, and smart cities, where urban infrastructure relies on intelligent decision-making, the versatility of heterogeneous multi-core processors becomes instrumental. They furnish the computational agility necessary to handle diverse workloads, optimizing performance and energy efficiency concurrently [1].

Within the expansive realm of heterogeneous multi-core processors, the ARM big.LITTLE architecture stands as a beacon of innovation. Forged by ARM Holdings, this architecture introduces a dynamic blend of high-performance and energy-efficient cores within a single System-on-Chip (SoC). The architectural prowess of ARM big.LITTLE lies in its adaptability—it intelligently allocates tasks to the most suitable core based on real-time processing demands. This dynamic allocation ensures optimal performance while conserving power, making it an ideal candidate for the diverse computing requirements posed by IoT devices and the intricate systems orchestrating smart cities [4]–[6].

The application of heterogeneous multi-core processors, notably represented by ARM big.LITTLE architecture, finds profound relevance in the expansive landscape of IoT. In IoT ecosystems, where sensors, actuators, and smart devices collaboratively generate and process real-time data, the need for efficient and responsive processing becomes paramount. ARM big.LITTLE's capability to seamlessly transition between high-performance and energy-efficient cores ensures that IoT devices can meet the demands of varying workloads, striking a delicate balance between computational power and energy conservation. This adaptability is essential for sustaining the growth and scalability of IoT applications.

While the computational prowess of heterogeneous multi-core processors is evident, the reliability of these systems becomes a critical consideration. In the context of IoT and smart cities, where uninterrupted functionality is paramount, ensuring the reliability of processors like ARM big.LITTLE is non-negotiable. The intricacies of unpredictable workloads, environmental factors, and the demand for real-time responsiveness underscore the imperative of reliability in these systems. This article undertakes the exploration of reliability evaluation in heterogeneous multi-core processors, shedding light on methodologies and strategies to fortify the robustness of these architectures, with a specific focus on ARM big.LITTLE [5], [7], [8].

In the Internet of Things (IoT) realm, failure-sensitive applications play a pivotal role in critical domains such as healthcare. Consider a healthcare IoT application that monitors and regulates medication delivery to patients with chronic conditions. In this scenario, the failure of IoT devices, such as sensors or drug infusion pumps, could severely affect patient health. For instance, a failure to accurately measure vital signs or administer medication could lead to incorrect diagnoses, ineffective treatments, or potentially life-threatening situations [8]. As these IoT applications become increasingly integrated into healthcare systems, ensuring their reliability and mitigating the risks associated with failures becomes imperative for patient safety and overall system effectiveness.

The ARM big.LITTLE processor, known for its energy-efficient architecture, is extensively utilized in healthcare IoT

979-8-3503-7467-4/24/$31.00 ©2024 IEEE

devices due to its ability to balance high performance and power efficiency. In healthcare IoT, these processors may be employed in monitoring devices, wearable health trackers, or portable diagnostic tools. Failure analysis becomes crucial for ARM big.LITTLE processors in healthcare IoT, as any malfunction or downtime could compromise the real-time processing of patient data, leading to delayed diagnoses or incorrect medical interventions. Understanding the root causes of failures, whether they stem from hardware issues, software bugs, or external factors, is essential for enhancing the reliability of these processors in healthcare applications. Moreover, failure analysis aids in developing strategies for fault tolerance, resilience, and system recovery, ensuring uninterrupted and accurate healthcare services in IoT ecosystems.

The paper is organized as follows, Section II meticulously delineates our simulation and test injection methodology, offering insights into the precision of our approach. Section III shows the fault injection results and a nuanced soft fault sensitivity analysis across diverse system configurations. In Section IV, the article culminates in our conclusions and forward-thinking ideas for future endeavors, providing a cohesive narrative that traverses the landscape of related works, methodological intricacies, empirical findings, and overarching implications.

II. Simulation and Fault Injection

Within this section, we provide a comprehensive exploration of our simulation and test injection methodology tailored for a meticulous assessment of the reliability of heterogeneous multi-core processors. Our specific focus centers on the acclaimed ARM big.LITTLE architecture, utilizing the ODROID-XU3 as a representative platform. This choice stems from the ODROID-XU3's integration of ARM Cortex-A15 and Cortex-A7 cores in a big.LITTLE configuration, offering a real-world emulation of computing scenarios encountered in IoT and smart city applications [9].

A. Platform Model

The ODROID-XU3 serves as the cornerstone for our reliability analysis, presenting a versatile board with indispensable features crucial for the evaluation of ARM big.LITTLE architecture. This board is equipped with two distinct clusters, each housing four ARM Cortex-A15 and Cortex-A7 cores, providing a balanced configuration for diverse computing workloads (Fig. 1.).

A pivotal aspect of the ODROID-XU3 is its support for Dynamic Voltage and Frequency Scaling (DVFS), a feature integral to the dynamic adjustment of voltages and frequencies in response to the system's workload. DVFS, tightly intertwined with operating frequencies, facilitates real-time adaptations that are instrumental in optimizing power consumption while preserving performance levels. This capability is especially significant in the context of reliability assessments for heterogeneous multi-core processors, where efficiency and responsiveness are paramount.

The ODROID-XU3's dual-cluster architecture, specific core configurations, and robust support for DVFS collectively establish it as a formidable platform for our in-depth reliability analysis. The inclusion of these features ensures a comprehensive exploration of the ARM big.LITTLE architecture's reliability, considering the nuanced dynamics introduced by varying workloads and dynamic adjustments facilitated by DVFS.

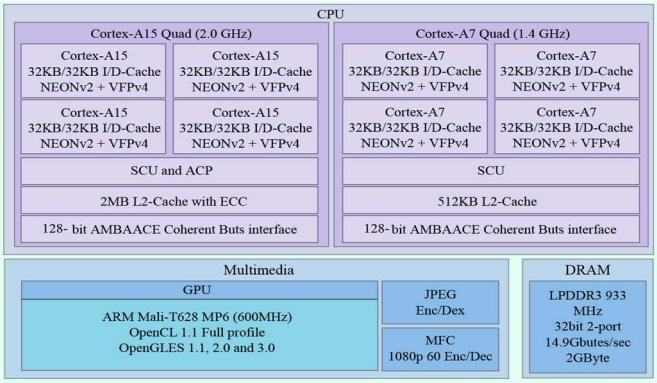

Fig. 1. ODROID-XU3 Diagram [10]

The gem5 simulator serves as a crucial tool in replicating the intricate behavior of the ARM big.LITTLE architecture within a controlled environment. Leveraging gem5's versatile capabilities, we can meticulously configure various parameters to ensure an accurate emulation of the ODROID-XU3 platform. This simulator proves invaluable in creating scenarios that allow for in-depth analysis of the system's response to diverse workloads and fault conditions. Our simulation methodology involves precision tuning of gem5 settings to closely mirror the intricacies of the selected platform. Notably, we have conducted simulations in two modes: bare metal and OS-based, aiming for a more comprehensive and detailed analysis of the system's performance and reliability under different conditions.

B. Benchmark

In our reliability analysis, we turn to the Mibench benchmark suite, a versatile set of workloads carefully selected for their representation of real-world application scenarios. Mibench encompasses a diverse range of benchmarks spanning automotive, consumer, and telecommunications applications, making it a comprehensive and representative suite for our evaluation.

Mibench workloads encapsulate various computing patterns, including signal processing, data compression, and automotive control applications. Each benchmark within the suite is designed to stress specific aspects of system performance, enabling a thorough examination of the ODROID-XU3 platform under diverse scenarios. These features make Mibench an ideal choice for our reliability analysis, ensuring that our evaluation covers a broad spectrum of workloads relevant to the complexities of IoT and smart city applications.

The selection of Mibench is particularly relevant in the context of IoT. IoT environments are characterized by the collaboration of diverse devices, each with unique computational demands. Mibench's diverse set of benchmarks allows us to simulate realistic workloads encountered in IoT scenarios, where devices must efficiently process data from various sensors, actuators, and smart devices. By incorporating Mibench in our reliability analysis, we ensure that our evaluation reflects the intricate computing requirements inherent in IoT ecosystems. This choice aligns with the broader goal of understanding the reliability of the ODROID-XU3 platform within the context of emerging technologies such as IoT and smart cities.

C. Fault Injection

In our reliability analysis, the introduction of controlled faults is paramount to assess the robustness of the ARM big.LITTLE architecture on the ODROID-XU3 platform under real-world conditions. Fault injection allows us to emulate various fault scenarios and observe the system's response, shedding light on its vulnerability and recovery mechanisms.

One of the fault injection techniques employed is the bit flip, a method where a bit within the system's memory or registers is intentionally toggled. This type of fault can simulate the impact of cosmic radiation, electrical noise, or other environmental factors leading to transient errors. In addition to bit flips, we explore various fault types, including transient and intermittent faults, mirroring the unpredictability encountered in dynamic computing environments. These fault types are essential to comprehensively evaluate the ODROID-XU3 platform's resilience under diverse fault conditions.

The incorporation of fault injection techniques is particularly relevant in the context of IoT. In IoT ecosystems, devices are exposed to diverse and often unpredictable environmental conditions. The reliability of processing units, like the ARM big.LITTLE architecture, is crucial in ensuring uninterrupted functionality in scenarios where IoT devices are deployed. Fault injection provides insights into how well the ODROID-XU3 platform can handle unexpected faults, making it a valuable tool for assessing reliability in the context of IoT applications. By simulating faults that may arise from external influences, we gain a deeper understanding of the ARM big.LITTLE architecture's robustness in sustaining reliable operations within the intricate landscape of IoT and smart cities.

The overarching goal of our simulation and fault injection endeavors is to evaluate the reliability of the ODROID-XU3 platform with the ARM big.LITTLE architecture. By subjecting the system to diverse workloads, environmental conditions, and fault scenarios, we aim to unravel its robustness. The insights gained from this comprehensive evaluation will inform strategies to enhance the reliability of heterogeneous multi-core processors, specifically focusing on ARM big.LITTLE architecture, in the context of IoT and smart city applications.

III. Experimental Results

In this section, we unveil the results stemming from our thorough evaluation of the ODROID-XU3 platform, harmonized with the ARM big.LITTLE architecture. Our assessment leveraged not only the MiBench benchmark suite but also Matrix multiplication scenarios, all orchestrated through the sophisticated gem5 simulator. These experimental findings offer profound insights into the platform's nuanced performance across diverse conditions and its adeptness in withstanding faults. This contribution adds substantial depth to our understanding of the platform's reliability within the intricate landscapes of IoT and smart city applications.

A. Fault Injection

For our fault injection methodology, we developed a software injector inspired by the approach outlined in [11]. In the course of our experiments, this injector introduces faults into the register bank of a designated core within the large cluster and another core within the small cluster. By changing one bit of the target register's value, the fault is simulated. These target registers encompass a spectrum of elements, spanning general-purpose registers to specific-purpose registers like PC and LR. Specifically, Cortex-A15 comprises 16 general-purpose 32-bit registers, while Cortex-A7 boasts 31 general-purpose 32-bit registers. As the fault injection process is identical for both cores, the fault injection is executed solely on the first 16 general-purpose registers of Cortex-A7, ensuring a consistent and controlled experimental environment.

In our injection experiments, we employed a selection of programs from the MiBench benchmark suite. This suite encompasses various programs tailored for assessing the performance of embedded systems, with a specific focus on applications pertinent to IoT devices. Also, we have used matrix multiplication:

- Bitcount: This program counts the number of set bits in an integer. It is a simple arithmetic operation, which can be relevant for IoT devices with low-power requirements.
- JPEG Compression (JPEG): IoT devices with image processing capabilities might benefit from evaluating the performance of JPEG compression.
- GSM Encryption (GSM): Assessing the performance of encryption algorithms like A5/1 used in GSM can be relevant for IoT devices that require secure communication.
- Patricia Tries (Patricia): IoT devices involved in networking and IP address lookup may find this program useful.
- FFT: Fast Fourier Transform is commonly used in signal processing applications, which may be relevant for certain IoT scenarios.
- Matrix multiplication benchmark: with matrices of 20 × 20 elements and data size of 32 bits.

The benchmarks employed for each core involve running the same application code on both types of cores, enabling a direct comparison of results to detect errors. Despite utilizing identical program code, it's crucial to note that each core operates with its instance of the program, residing in a dedicated memory section. This distinction implies that, even though the program code remains consistent, each core accesses different memory addresses within the shared memory. This architectural setup ensures that the evaluation captures any divergences or errors that may arise due to the unique memory interactions of each core, providing a comprehensive assessment of the platform's reliability under varied conditions.

Coresight, stands as an advanced debugging and tracing technology seamlessly integrated into the architecture of both large and small cores. This pivotal feature plays a crucial role in significantly augmenting the platform's capabilities for debugging and trace operations [7], [12]. Coresight offers valuable insights into the intricate internal system behavior, particularly during fault injection tests. Its seamless integration contributes to the comprehensive understanding of system dynamics, aiding in the identification and analysis of issues that may arise under fault conditions. Figure 2 visually underscores the strategic placement and functionality of Coresight within the architecture, emphasizing its integral role in enhancing the system's diagnostic capabilities.

B. Data Flow Error

Data Flow Error is a category of error that pertains to discrepancies or alterations in the flow of data within a system [13], [14]. It encompasses instances where the expected or actual data movement deviates from the established patterns or protocols Throughout our extensive evaluation, deliberately injecting over 5000 errors into the register bank for each benchmark unveiled the nuanced effects of data errors. On average, approximately 7% of injected faults manifest as errors in the large core, while around 10% result in errors in the small core. This discrepancy underscores the substantial impact of Coresight in adeptly recording and analyzing the system's response to errors. Additionally, it highlights differences in the maximum execution frequency of each core type, contributing to a comprehensive understanding of how data errors influence the reliability of the ARM big.LITTLE architecture.

In TABLE I**.** a comprehensive summary unfolds, detailing the experimental fault injection results in the general-purpose register under the bare metal simulation mode. This table meticulously outlines the fault-to-error conversion for all benchmarks, providing essential insights into the system's behavior under fault conditions. Additionally, it includes crucial information about the kernel responsible for generating each error. Significantly, a discernible discrepancy in the total number of errors surfaces between the two types of kernels for each benchmark. This variance in error occurrences illuminates each core type's nuanced behavior and distinctive characteristics, offering valuable insights into their respective responses to fault injections.

TABLE I. BARE METAL INJECTION FOR DFE

Type of core	Fault injection		
	Benchmark	***Number of Fault***	***Undetected***
Cortex-A15	Bitcount	5029	328 (6.52%)
	JPEG	5038	362 (7.18%)
	GSM	5126	353 (6.88%)
	Patricia	5019	341 (6.79%)
	FFT	5056	348 (6.88%)
	MM	5108	372 (7.28%)
Cortex-A7	Bitcount	5029	498 (9.9%)
	JPEG	5038	537 (10.65%)
	GSM	5126	522 (10.18%)
	Patricia	5009	519 (10.36%)
	FFT	5056	520 (10.28%)
	MM	5108	551 (10.78%)

TABLE II**.** presents a comprehensive summary delineating the experimental results of error injection in the general-purpose register under OS-based simulation mode with Ubuntu 18.04 operating system. This table systematically outlines the fault-to-error conversion for all benchmarks, providing crucial insights into the system's behavior under fault conditions. Additionally, it includes pertinent information about the kernel responsible for generating each error. A discernible difference in the total number of surfaced errors is evident between the two kernel types for each criterion, shedding light on the nuanced behavior and distinctive features inherent in each core type. This information offers valuable insights into their respective responses to fault injection, contributing to a comprehensive understanding of the system's reliability under varied conditions in the simulation mode based on the Ubuntu 18.04 operating system.

C. Control Flow Error

An aspect of fault-to-error conversion is Control Flow Error (CFE), a type of error that deviates from the primary system flow [11], [15]. To enhance our understanding of the reliability of these systems, deliberate error injections were performed using the following methods, enabling the evaluation of system resilience to CFE:

- Injecting a Bit-Flip in PC and LR Registers: Simulating faults by inducing bit-flips in critical registers, specifically the Program Counter (PC) and Link Register (LR), to observe the impact on the system's control flow.
- Injecting a Bit-Flip in a Branch Instruction: Deliberately introducing bit-flips within branch instructions to assess the system's response to altered control flow conditions.

TABLE II. OS-BASED INJECTION FOR DFE

Type of core	Fault injection		
	Benchmark	***Number of Fault***	***Undetected***
Cortex-A15	Bitcount	5087	297 (5.83%)
	JPEG	5013	322 (6.42%)
	GSM	5109	318 (6.22%)
	Patricia	5108	309 (6.04%)
	FFT	5019	314 (6.29%)
	MM	5099	327 (6.41%)
Cortex-A7	Bitcount	5087	431 (8.47%)
	JPEG	5013	463 (9.23%)
	GSM	5109	452 (8.84%)
	Patricia	5108	449 (8.79%)
	FFT	5019	453 (9.02%)
	MM	5099	472 (9.25%)

- Replacing a Branch Instruction with a Non-Branch Instruction: Experimenting with the substitution of a branch instruction with a non-branch instruction, simulating a scenario where the expected control flow is intentionally disrupted.
- Removing a Branch Instruction: Deliberately eliminating a branch instruction to gauge the system's adaptability and resilience when faced with a sudden absence of expected branching.

To scrutinize the effect of control flow errors, the aforementioned fault injection techniques were employed to simulate scenarios where control flow errors are intentionally manipulated. More than 5000 faults were injected for each benchmark, allowing for a comprehensive investigation into how the system responds to deliberate alterations in control flow, providing valuable insights into its resilience and adaptability.

TABLE III**.** presents a comprehensive summary of the experimental results for control flow error injection in the previously mentioned scenarios in bare metal simulation mode. This table meticulously captures the fault-to-error conversion for all metrics, offering essential insights into the system's behavior under fault conditions. Crucially, it includes pertinent information about the kernel responsible for generating each error. A noticeable disparity in the total number of surfaced errors emerges between the two kernel types for each criterion. This discernible variation in fault occurrences sheds light on the nuanced behavior and distinctive features inherent in each core type, providing valuable insights into their respective responses to fault injection. presents a comprehensive summary of the experimental results for control flow error injection in the previously mentioned scenarios.

In TABLE IV**.** provides a comprehensive summary of experimental results for control flow error injection in the previously mentioned scenarios under OS-based simulation mode. This table accurately shows the fault-to-fault conversion for all metrics and provides essential insights into system behavior under fault conditions.

TABLE III. BARE METAL INJECTION FOR CFE

Type of core	Fault injection		
	Benchmark	***Number of Fault***	***Undetected***
Cortex-A15	Bitcount	5012	345 (6.88%)
	JPEG	5160	372 (7.2%)
	GSM	5138	381 (7.41%)
	Patricia	5201	361 (6.94%)
	FFT	5018	368 (7.33%)
	MM	5098	391 (7.66%)
Cortex-A7	Bitcount	5012	512 (10.21%)
	JPEG	5160	531 (10.29%)
	GSM	5138	564 (10.97%)
	Patricia	5201	529 (10.17%)
	FFT	5018	538 (10.72%)
	MM	5098	576 (11.29%)

D. Reliability Analyse

In this section, we delve into the comprehensive analysis of the reliability of the ODROID-XU3 platform equipped with ARM big.LITTLE architecture, building upon the experimental results and insights garnered from fault injections, data flow errors (DFE), and control flow errors. (CFE).

The capabilities of the tested board reveal a notable observation – many faults do not transition into errors, highlighting a positive aspect of ARM big.LITTLE systems.

This resilience to error manifestation underlines the robustness of the architecture in mitigating the impact of various faults. However, a crucial determinant in the fault-to-error conversion process is the presence or absence of the operating system. This factor significantly influences whether a fault evolves into an error, emphasizing the intricate interplay between hardware and software components in shaping the reliability dynamics of the ARM big.LITTLE architecture.

TABLE IV. OS-BASED INJECTION FOR CFE

Type of core	Fault injection		
	Benchmark	***Number of Fault***	***Undetected***
Cortex-A15	Bitcount	5019	298 (5.93%)
	JPEG	5113	309 (6.04%)
	GSM	5029	314 (6.24%)
	Patricia	5201	319 (6.13%)
	FFT	5167	340 (6.58%)
	MM	5128	349 (6.8%)
Cortex-A7	Bitcount	5019	465 (9.26%)
	JPEG	5113	479 (9.36%)
	GSM	5029	477 (9.48%)
	Patricia	5201	496 (9.53%)
	FFT	5167	499 (9.65%)
	MM	5128	512 (9.99%)

Error Detection Rate (EDR) is a critical reliability analysis factor that quantifies the effectiveness of a system in identifying and detecting errors. It provides a quantitative measure of the platform's capability to recognize faults or anomalies within its operation. In the context of the ODROID-XU3 platform with ARM big.LITTLE architecture, understanding the Error Detection Rate involves assessing how well the system can identify errors resulting from intentional fault injections, data flow errors (DFE), and control flow errors (CFE).

The EDR is typically expressed as a percentage and is calculated using the formula [7], [16]:

$$EDR = \left(\frac{Number\ of\ Detected\ Errors}{Total\ Number\ of\ Injected\ Faults}\right) \times 100 \qquad (1)$$

Here, the "Number of Detected Errors" represents the count of errors successfully identified by the platform during the experimental evaluation. The "Total Number of Injected Faults" corresponds to the intentional faults or errors injected into the system for testing purposes. A higher Error Detection Rate indicates a more robust and reliable system, as it implies that a significant proportion of injected faults were successfully detected. Conversely, a lower EDR may indicate potential vulnerabilities in error detection mechanisms, highlighting areas for improvement in the system's reliability.

In Fig. 2., we present a comprehensive evaluation of the platform's reliability using the EDR factor. This evaluation involves the systematic analysis of intentional fault injections, data flow errors (DFE), and control flow errors (CFE). The x-axis of Figure 1 represents different fault injection scenarios and benchmarks, while the y-axis signifies the Error Detection Rate as a percentage. Each data point on the graph corresponds to a specific condition, providing a visual representation of how well the platform identifies errors under various circumstances.

Based on the results depicted in Fig. 2., several key conclusions can be drawn. The analysis underscores that, to

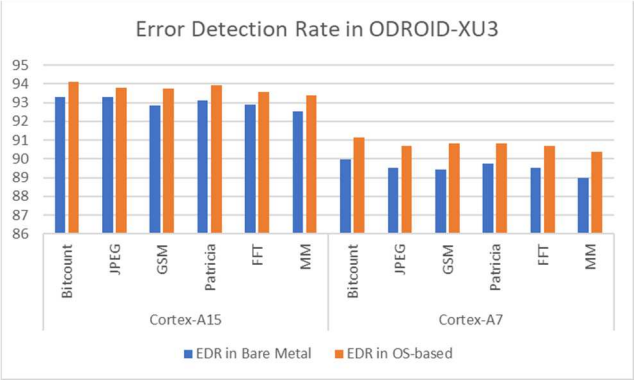

Fig. 2. Error Detectin Rate Analysis

minimize power consumption and ensure reliability, ARM big.LITTLE architectures serve as apt choices for IoT systems. The ability to strategically map tasks onto different clusters based on their importance and workload nature contributes to the overall efficiency and dependability of the system.

This conclusion aligns with the observation that the ODROID-XU3 platform is equipped with ARM big.LITTLE architecture, exhibits resilience and effectiveness in handling intentional fault injections. The Error Detection Rate serves as a valuable metric for quantifying the platform's robustness under varied conditions, providing insights that can inform strategic decisions in IoT and diverse computing applications. The platform's heterogeneous nature and adaptability across different benchmarks highlight its suitability for addressing the challenges of reliability and energy efficiency in contemporary computing scenarios.

IV. Conclusion and Future Work

In summary, our investigation into the dynamic landscape of the Internet of Things (IoT) has brought to light the escalating demand for robust processing solutions, propelling the widespread adoption of heterogeneous multi-core processors, exemplified prominently by the ARM big.LITTLE architecture. With a targeted focus on critical sectors such as healthcare, where the reliability of failure-sensitive applications takes precedence, our study accentuates the vital need for tailored failure analysis specific to ARM big.LITTLE processors. This study is a thorough reliability analysis of the ODROID-XU3 platform under diverse conditions. Employing deliberate fault injections, evaluating data flow errors (DFE), scrutinizing control flow errors (CFE), and considering the Error Detection Rate factor collectively underscore the platform's resilience and adaptability. Particularly noteworthy is the ARM big.LITTLE architecture's effective fault management with minimal power consumption, positioning it as a promising solution for Internet of Things (IoT) systems. Our findings underscore the significance of strategic task mapping across different clusters based on workload characteristics, contributing significantly to heightened reliability and efficiency in computing environments.

Future work may delve into exploring the scalability of ARM big.LITTLE architectures for larger systems, examining real-world applications to broaden the scope of performance analysis. Further investigation into advanced fault injection techniques and the integration of additional metrics could refine the evaluation process. Continuous refinement of fault tolerance mechanisms and exploration of adaptive strategies based on dynamic workload characteristics will advance the reliability and performance of ARM big.LITTLE architectures in evolving computing paradigms. The ongoing pursuit of these directions will contribute to a deeper understanding and optimization of heterogeneous computing platforms.

References

[1] V. Tambe, G. Bansod, S. Khurana, and S. Khandekar, “Reliability and availability of IoT devices in resource constrained environments,” International Journal of Quality and Reliability Management, pp. 1648–1662, 2022.

[2] S. Tian, W. Ren, Q. Deng, S. Zou, and Y. Li, “A Predictive Energy Consumption Scheduling Algorithm for Multiprocessor Heterogeneous System,” IEEE Transactions on Green Communications and Networking, pp. 979–991, 2022.

[3] U. Ahmed, J. C. W. Lin, and G. Srivastava, “Heterogeneous Energy-aware Load Balancing for Industry 4.0 and IoT Environments,” ACM Transactions on Management Information Systems, 2022.

[4] C. Wang, X. Yu, L. Xu, and W. Wang, “Energy-Efficient Task Scheduling Based on Traffic Mapping in Heterogeneous Mobile-Edge Computing: A Green IoT Perspective,” IEEE Transactions on Green Communications and Networking., pp. 972–982, 2023,

[5] M. N. M. Najib and D. A. Ramli, “Analysis of Smart IoT Portal Based on Advanced RISC Machines (ARM) Processor for Fanless Heat Maintenance, ” Lecture Notes in Electrical Engineering. Springer Singapore, 2022.

[6] M. Z. Khan, O. H. Alhazmi, M. A. Javed, H. Ghandorh, and K. S. Aloufi, “Reliable internet of things: Challenges and future trends,” Electronics, pp. 1–22, 2021.

[7] M. Pena-Fernandez, A. Lindoso, L. Entrena, M. Garcia-Valderas, Y. Morilla, and P. Martin-Holgado, “Online error detection through trace infrastructure in ARM microprocessors,” IEEE Transactions on Nuclear Science, 2019.

[8] N. Taimoor and S. Rehman, “Reliable and Resilient AI and IoT-Based Personalised Healthcare Services: A Survey,” IEEE Access , pp. 535–563, 2022.

[9] G. S. Rodrigues, F. Rosa, Á. B. De Oliveira, F. L. Kastensmidt, L. Ost, and R. Reis, “Analyzing the Impact of Fault-Tolerance Methods in ARM Processors under Soft Errors Running Linux and Parallelization APIs,” IEEE Transactions on Nuclear Science, pp. 2196–2203, 2017.

[10] ODROID-XU3 Hardware. (2017). Retrieved from https://wiki.odroid.com/old_product/odroid-xu3/hardware/xu3_hardware

[11] H. A. H. Ahmad, Y. Sedaghat, and M. Moradiyan, “LDSFI: A lightweight dynamic software-based fault injection,” 2019 9th International Conference on Computer and Knowledge Engineering (ICCKE), pp. 207–213.

[12] S. M. A. Zeinolabedin, J. Partzsch, and C. Mayr, “Real-Time Hardware Implementation of ARM CoreSight Trace Decoder,” IEEE Design and Test, pp. 69–77, 2021.

[13] A. Lindoso, M. Garcia-Valderas, and L. Entrena, “Analysis of neutron sensitivity and data-flow error detection in ARM microprocessors using NEON SIMD extensions,” Microelectronics Reliability, p. 113346, 2019.

[14] M. Didehban, H. So, P. Gali, A. Shrivastava, and K. Lee, “Generic Soft Error Data and Control Flow Error Detection by Instruction Duplication,” IEEE Transactions on Dependable and Secure Computing, pp. 78–92, 2024.

[15] H. A. H. Ahmad and Y. Sedaghat, “Software-based Control-Flow Error Detection with Hardware Performance Counters in ARM Processors,” Proceedings - 2022 CPSSI 4th International Symposium on Real-Time and Embedded Systems and Technologies (RTEST).

[16] P. M. Aviles, A. Lindoso, J. A. Belloch, and L. Entrena, “Evaluating reliability through soft error triggered exceptions at ARM Cortex-A9 microprocessor,” Microelectronics Reliability, p. 114323, 2021.