Run Time Memory Error Recovery Process in Networking System

Carlo Vitucci
Technology Management
Ericsson AB
Stockholm, Sweden
carlo.vitucci@ericsson.com

Daniel Sundmark
Computer Science and Software Enigineering
Mälardalen University
Västerås, Sweden
daniel.sundmark@mdu.se

Jakob Danielsson
Sys Architecture
Ericsson AB
Stockholm, Sweden
jakob.danielsson@ericsson.com

Marcus Jägemar
Sys Compute Dimensioning
Ericsson AB
Stockholm, Sweden
marcus.jagemar@ericsson.com

Alf Larsson
Senior Specialist Observability
Ericsson AB
Stockholm, Sweden
alf.larsson@ericsson.com

Thomas Nolte
Division of Networked and Embedded System
Mälardalen University
Västerås, Sweden
thomas.nolte@mdu.se

***Abstract*—System memory errors have always been problematic; today, they cause more than forty percent of confirmed hardware errors in repair centers for both data centers and telecommunications network nodes. Therefore, it is somewhat expected that, in recent years, device manufacturers improved the hardware features to support hardware-assisted fault management implementation. For example, the new standard, DDR5, includes both data redundancy, the so-called Error Correcting Code (ECC), and physical redundancy, the post-package repair (PPR), as mandatory features. Production and repair centers mainly use physical redundancy to replace faulty memory rows. In contrast, field use still needs to be improved, mainly due to a need for integrated system solutions for network nodes. This paper aims to compensate for this shortcoming and presents a system solution for handling memory errors. It is a multi-technology proposition (mixed use of ECC and PPR) based on multi-layer (hardware, firmware, and software) error information exchange.**

***Index Terms*—Memory Faults, Fault Management, Post-Package Repair, Error Correcting Code, Run Time Fault Recovering**

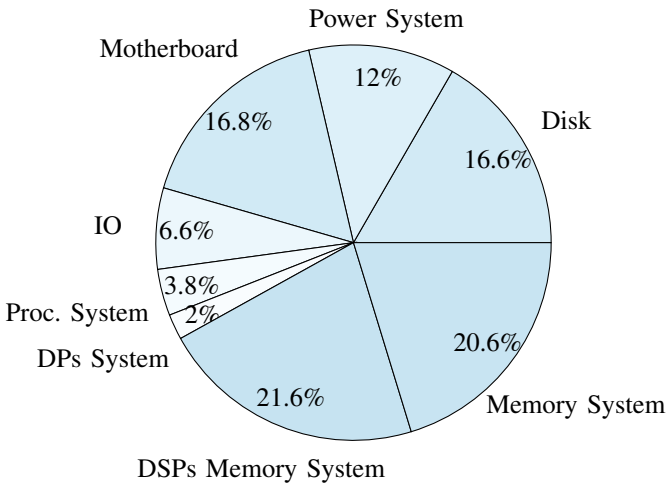

Fig. 1. Fault ranking in a telecom node

I. Introduction

Miniaturization of hardware components and higher density of transistors per component has enabled significant gains in performance, energy efficiency, and product cost [1]. Unfortunately, the miniaturization process compromises the reliability of the integrated circuits. The leading cause of this reliability loss is that the nanoscale technology has reduced the critical charge required for a logic-level switch in the device [2]. The stringent limits on power consumption, and the consequent voltage reduction, has lead to an increase in error probability. The most significant risk is data corruption. As a consequence of higher probability of data corruption, memory devices in networking nodes are the most prone to hardware errors [3]. Fig. 1 (fault ranking statistics for telecommunication nodes) and Fig. 2 (fault ranking statistics for data center nodes [3]) show how memory devices are the primary source of hardware errors.

DDR5 memories are an excellent example of how hardware component manufacturers are well aware of the decreased reli-

Acronym	Explanation
BIST	Build-in Self-Test
CAE	CorrectAble Error
CE	Corrected Error
ECC	Error Correcting Code
hPPR	Hard PPR, permanent PPR
MBIST	memory BIST
mPPR	MBIST PPR, available in BIST mode
PRR	Post Package Repair, row swap memory device capability
sPPR	Soft PPR, temporary PPR
UCE	UnCorrectable Error

TABLE I
Acronym table

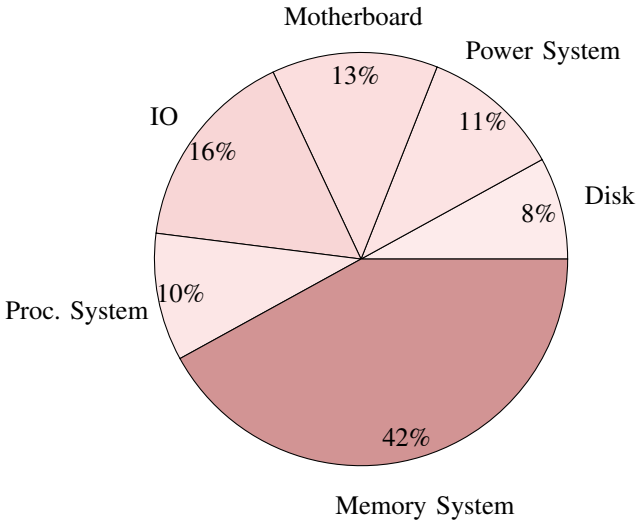

Fig. 2. Fault ranking in data center node [3]

ability of the device. One of the main objectives of the DDR5 specification is to increase reliability using error detection and correction techniques [4]. The DDR5 specification includes two particularly interesting techniques: the extension of data redundancy with the introduction of on-die Error Correcting Code (ECC) and the extension of physical redundancy with the mandatory implementation of Post-Package Repair (PPR). ECC is an excellent mechanism that can both detect and correct memory errors during runtime, while PPR is an excellent instrument for recovering from a faulty memory area. The new system memory technology confirms the trend towards hardware miniaturization: DDR5 devices, with a higher transistor density than the previous DDR4, are increasingly used to count on more capacity and more outstanding performance (see Table II).

A previous work [6] designed a multi-layer and multi-technology fault management approach to increase product resiliency. The central concept is managing a fault recovery through all four layers of a networking system (hardware, firmware, software, and control system), each using the available recovery technologies for it and interchanging data with the other layers. The proposed fault management approach is a valuable framework to handle the system memory faulty condition too.

Features	**DDR4**	**DDR5**
Speed Clock	1.6 to 3.2 GT/s 0.8 to 1.6 GHz	4.4 to 6.4 GT/s 2.2 to 3.8 GHz
IO Voltage	1.2V	1.1V
Power Management	On Motherboard	On DIMM PMIC (Power Management IC)
Max. Die Density	16Gb	64Gb
ECC-on-die	NO	YES
PPR	Optional	Mandatory

TABLE II
FEATURES COMPARISON: DDR4 VS DDR5. [4], [5]

Paper Contribution

This paper describes a novel methodology for fault analysis and correction in system memory based on multiple redundancy techniques. The proposed method is deploying the fault management multi-layer framework in both hard and soft error based on multi-technology interwork: data (ECC) and physical (PPR) redundancies.

II. DEFINITIONS

This section recalls a set of definitions and concepts for the reader. These definitions differ slightly in standardization references based on the field of interest (automotive, nuclear, avionics, etc.).

***Definition** 1: ECC* stands for *Error Correcting Code*. It is an algorithm introduced by Hamming [7]. The algorithm introduces a few extra bits to encode data information. Decoding those redundancy bits allows for detecting a limited number of errors.

***Definition** 2: on-die ECC* is an implementation scheme for the ECC algorithm. On a write command, DRAM device computes ECC internally and stores the ECC code in additional reserved storage. On a read command, DRAM device reads back ECC code and can correct any single-bit error. By definition, on-die ECC doesn't provide any protection against error in the memory bus. It is available for DDR5 only [8].

***Definition** 3: side-band ECC* is an implementation scheme for ECC algorithm. The Memory controller generates, writes, and reads the ECC code side-band along with the actual data to the memory. It is available for DDR4 and DDR5 [8].

***Definition** 4: patrol scrub ECC* is a deterministic fault-tolerance technique that regularly repairs each faulty memory location at intervals to prevent memory from accumulating a second error in a single word. Consequently, the likelihood of uncorrectable error due to soft error remains as low as possible. [9].

***Definition** 5: PPR* stands for *Post-Package Repair*. It is a standard industrial capability defined by JEDEC [5]. It allows swapping degraded memory rows with spare one per bank group.

***Definition** 6: mPPR* stands for *MBIST PPR*. It is a dedicated set of memory rows, separated by standard PPR ones that Built-in Self-Test can use to swap degraded memory rows during the Advanced Memory Test execution [4]. This feature depends on the vendor solution because it is still optional in DDR5. It is not available in DDR4.

***Definition** 7: sPPR* stands for *Soft* PPR. It is a temporary post-package repair, volatile to power cycling [10].

***Definition** 8: hPPR* stands for *Hard* PPR. It is a permanent post-package repair, nonvolatile to power cycling [10].

***Definition** 9:* A *Corrected error* (CE) is a fault detected, located, and corrected within a single layer [11]. ECC-Single-error-correction is an example of algorithm able to detect, locate and correct a bit-flip in memory.

***Definition** 10:* A *Correctable error* (CAE) is a fault detected and located by a layer but evaluated as correctable by a

different layer performing one or multiple actions to recover the fault [11]. ECC-Single-Error-Detection is an example.

***Definition** 11:* A *Uncorrectable error* (UCE) is a fault detected and located by a layer and not recognized as correctable [11]. ECC-Double-Error-Detection is an example.

***Definition** 12:* A *Hard Error in memory* is single or multiple bits permanently stacked on a voltage level [12]. An electrostatic discharge (ESD), an electrical over-current or over-temperature condition, or fabrication or module assembly irregularities can cause a hard error.

***Definition** 13:* A *Soft Error in memory* is single or multiple bits temporally flipped [12]. An electrical disturbance, high-energy particle strikes, or electrical noise in the circuits can cause a soft error. It is helpful to mention the so-called "rowhammer effect", a charge disturbance between cells in the high-dense memory circuits.

***Definition** 14: BIST* stands for *Built-in Self-Test*. It is a device feature that consists of internal test execution without the need for an external tester [13]. It is called MBIST When the target of the self-test is the memory device(s) [14].

III. Related works

The introduction of new hardware-assisted fault management features has increased interest in the benefits and use of these redundancy techniques. For example, Cha et al. [1] and Khan et al. [2] investigated the negative effect of the device miniaturization and the importance of data redundancy to detect and correct the soft error in run-time. Other researchers also analyzed physical redundancy techniques and their integration in the Self-Test procedure (BIST) as hard error recovery mechanisms [15]. Jeddeloh [16] proposed a mechanism based on the error memory map concept, while Wada et al. [15] and Querbach et al. [17] focused on PPR integration in the BIST. Lee et al. [18] used the content-addressable memories method in hybrid with ECC to implement self-repair. However, since the PPR begins to be a mandatory feature with DDR5 technology, a more efficient solution should consider the integration between ECC and PPR. Lu et al. [19] and Manasa et al. [20] proposed a method to integrate PPR and ECC in production tests, while Kim and Milor [21], [22] represented the first attempt to integrate ECC and PPR in run time. All the solutions mentioned above have two critical problems: they require a modification of the memory device to determine the "hard" or "soft" nature of the error, and they always execute some device's commands for verifying the type of error, which implies a long execution time because the verification is done every time, even when not necessary. A multi-layer solution overcomes those critical issues. Chao et al. [23] foresee that the BIOS analyzes the memory state and uses the PPR to ban defective raws from being communicated to the OS to form the memory maps. The problem with this solution is the one-way communication which requires the execution of advanced memory tests at each boot, with longer restart times, by a magnitude of several minutes. Four patents aim to provide efficient integration of data and physical redundancy features in run-time. Pope [24] proposed to identify the nature of the memory error during the ECC exception handling and, if necessary, correct it via PPR. He suggested running the memory test and swap execution as part of the exception handling. Consequently, the exception time increase. A circumstance that this paper identified as undesirable behavior for radio access boards. Lee [25] and Muthilayu [26] proposed a similar solution: a memory fault exception set by ECC procedure is the trigger to migrate into BIST mode and manage PPR mechanism, But their design also implies a too-long exception time. Zimmer's proposal [27] is similar to the solution of the others. Still, it is specific to the Intel platform: it suggests the transition from OS to System Management Mode (SMM), typical of the x86 platform. It also inherits the longer exception handling time. Both Pope [24] and Muthilayu [26] mentioned the exchange of error information between software and firmware, even if they miss to clarify the terms and scope of this exchange of information.

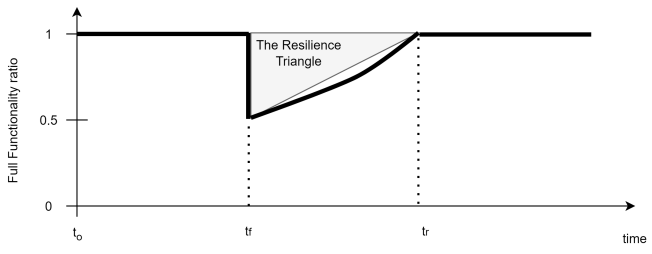

Fig. 3. The Resilience Triangle

The paper proposal, as described in Section V, is a valid alternative to previously proposed memory error recovering procedures when long times for the exception are more problematic than a controlled restart of the board.

IV. Research Description

A. Research Objective

The research aims to integrate data redundancy (ECC) and physical redundancy (PPR) in Radio Access Network (RAN) boards that use DDR4/DDR5 system memory devices. The integration of the fault redundancy mechanisms mentioned above will optimize the area of the resilience triangle [28] by minimizing recovery times from an error condition (see Fig. 3, the transition from a time when the fault occurs, t_f, to time when the system returns to its fully operational working mode, t_r).

Minimization is possible by interworking the different layers (hardware, firmware, software, and control system). Each layer performs its recovery attempt and, in case of success, completes the recovery procedure immediately, without invoking the higher layer if not necessary. Consequently, the system returns to the fully working operational mode in a time that can be lower than or equal to the time spent by the control system alone to recover from the fault. Since any layer passes the

outcome of the recovery action to that above, implementing the recovery mechanism per layer is minimal and optimal because there is no need to duplicate a procedure that a lower layer has performed already (and that a higher layer will then execute).

B. Context Description

The paper analyzes the management of system memory faults in the hardware infrastructure of networking embedded systems. More specifically, the study focuses on system memory fault in infrastructure for 5G networks [29] and beyond [30]. There are some characteristics of 5G network RAN nodes that are relevant to this paper:

- For cost optimization reasons, a 5G RAN node usually has limited storage availability, which could reduce the data collection capability.
- The very strict requirement of the fronthaul latency [31] for functional splitting [32] limits the computing resource available for data handling.
- 5G RAN nodes are interconnected to other network functions and entities in a service-based interface scenario. The exception time needs to be limited, or it would cause a reaction from the access and mobility management function [33].

Fig. 4 depicts a high-level infrastructure view of a typical 5g compute node, which we call "reference architecture". We describe the different components of the reference architecture in detail as follows:

- A host system that contains:
  - a processing unit capable of running firmware and software;
  - a memory controller that executes PPR logic, the side-band ECC, and the access logic to the memory device.
- A memory subsystem:
  - a memory controller, if not already supported by the host system and with the features already described for the host system;
  - a memory device with support for PPR (mandatory) and on-die ECC (optional, not directly required by our method, but helpful in keeping the probability of a soft error low);
  - a persistent memory area where to store the interwork data.

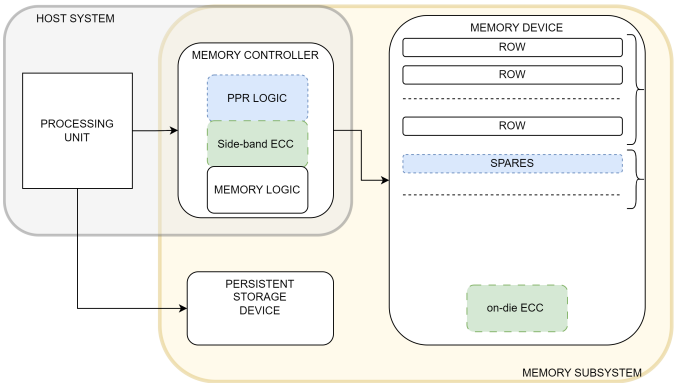

Fig. 4. The System architecture

V. Memory Error Handling Process

Firmware and software work in collaboration to obtain efficient memory error handling. The former manages the physical redundancy, the latter the data redundancy, and they share ECC error and PPR availability information using a persistent memory area that survives a processor restart (interwork data area). Software and firmware can also access another persistent memory area to log the system's hardware errors. Both firmware and software know about restart type; that is, they can recognize a board power-on. In the interwork data area, firmware creates and initializes two structures, PPR INFO and ECC INFO, at a board power-on. The PPR INFO contains the total number of hPPRs available in the system and the number of hPPRs currently in use. The ECC INFO includes a list of ECC faulty events, where each event contains the location (memory address where ECC detects an error), persistence level, and the number of occurrences of that specific error message.

A. Process in the Software Domain

Fig. 5 describes the memory error handling flow in the software domain. The software runs the ECC algorithm through the memory controller. Parallel fault recovery mechanisms, such as patrol scrubbing and on-die ECC, are unchanged and are strongly recommended as a method of immediate detection and correction of a soft error. When the ECC algorithm detects an error indication, the error handling process analyzes the typology. No other intervention is required if the hardware can automatically correct the error. If software actions can correct the fault, it is crucial to understand whether the suspect memory segment belongs to data or control memory. The software immediately carries out a write-back test in case of data memory corruption, as the memory contents cannot compromise the system's integrity. In the case of control memory, the error handling process applies the same policy as for uncorrectable errors: software checks if a valid entry exists in the ECC INFO table. If a non-validity entry exists, the software will create an entry with correct validity, non-persistence, and non-occurrence indication. If it exists, it just increases its occurrence.

B. Process in the Firmware Domain

Fig. 6 describes the memory error handling flow in the firmware domain. After a processor restart, the firmware considers whether ECC error structures are available in the ECC INFO table and evaluates for each entry the validity, persistence, and occurrence. The firmware continues its board initialization phase if there is no fault listed in the interwork data area. For valid entries, the procedure processes only those with persistence and multiple occurrences and swap to an hPPR, if available. The procedure considers an indication as

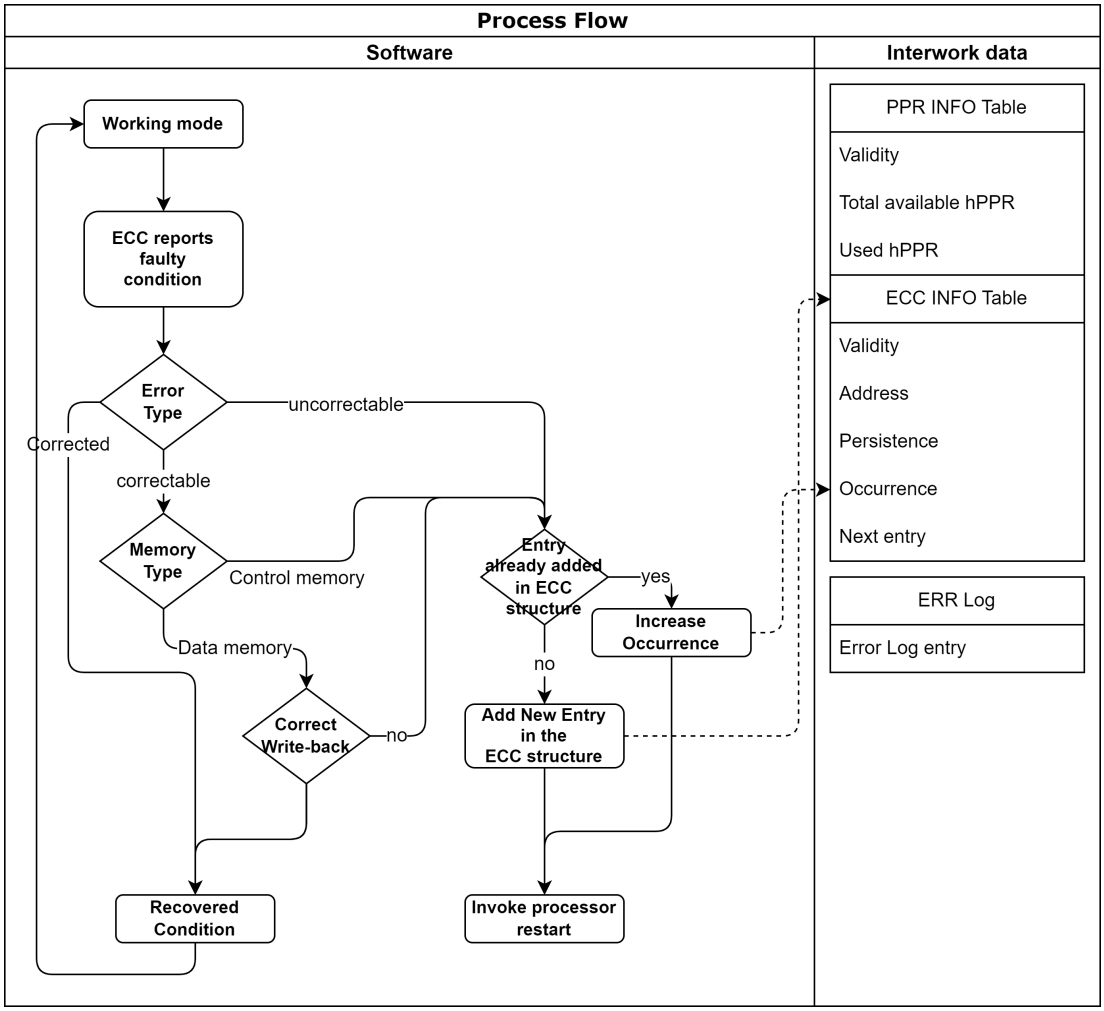

Fig. 5. Visualization flow of the software Memory Error handling Proposal

persistent if it is present in the ECC INFO for two consecutive processor restarts. An indication results in multiple occurrences if the ECC of the memory controller reports an error on the exact location after a second processor restart. Two cases deserve further description.

A permanent and multiple occurrences error is a ”hard error” because, by definition, a simple write-back test cannot work. The occurrence of a soft error on the exact memory location, even after a processor restart, is a rare event. Critical working conditions, such as a high-temperature condition, insufficient power, or incorrect device initialization, are more likely. Therefore, it is helpful to report this suspected erroneous environmental condition in the hardware error log.

The case where all hPPR spares are already in use should also add an entry in the hardware error log since firmware and software can no longer recover from the fault condition.

C. The following items listed the key features that distinguishes the paper proposal to other researches

.

- **Memory error handling using ECC and PPR integration for radio access network boards in run-time**. The multi-technology integration allow recovery action for memory error in case of hard error. Whenever the software detects an uncorrectable fault, the method considers that it is safer to proceed with a recovery attempt through a processor restart. Blocking the system to handle the exception in the firmware domain would introduce too long times that the network node control system would not tolerate [32], [33]. The proposed method uses other fault management features, such as patrol scrub, on-die ECC, and poisoning, to minimize the risk that a soft error causes an uncorrectable fault.
- **Minimal restart time**. Firmware and software share error information using interwork data. For that reason, firmware doesn’t need to run full memory integrity test to detect possible memory rows containing hard error (that can takes even minutes). The firmware executes only the bare minimum of write-back tests, exclusively on memory rows reported as defective by the software and only in the case of multiple error occurrences. In

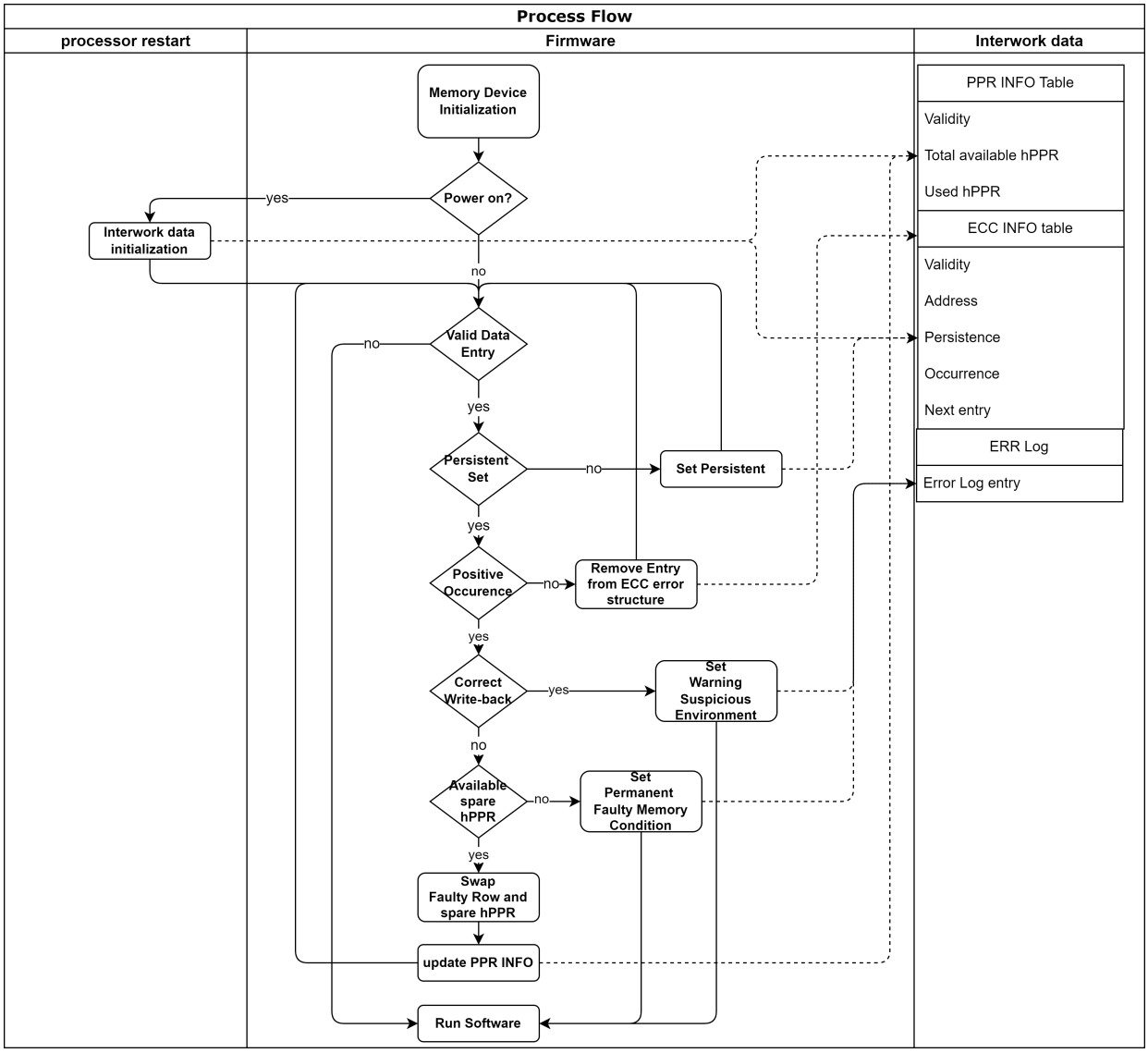

Fig. 6. Visualization flow of the Firmware Memory Error handling Proposal

practice, the PPR swap procedure, the more expensive function (in terms of execution time) is used exclusively in the presence of a hard error.

- **Integrated solution** The solution is based on hardware, firmware, and software fault management mechanisms fully supported by the most important vendors of components for embedded systems, which makes it attractive because the implementation doesn't require special hardware, hardware adapters. Moreover, the same design cover the production test needs, avoiding the maintenance of a separated "production-only" software.
- **Optimal usage of share resources** The swap procedure for hPPR is performed only in case of a hard error.

VI. The PPR technology usage in Production test

Production and repair tests are two moments of a product's life already using PPR technology [34]. PPR usage is possible because the limitation described in section IV-B are valid only in run-time. Of course, production tests can only use the PPR function if supported by the hardware. But with the migration to the DDR5 technology (see Table II), the mandatory availability of ECC and PPR is a concrete opportunity to have a shared memory handling solution for all production's life phase (from production to run-time, from run-time to repair center). A common procedure removes the need for "ad hoc" production test software, reducing development time and maintenance costs. For this reason, it is worth analyzing how the production test work and how it manages a memory error condition.

The production test executes a sequence of threads called test-case. The System memory device test-case enables the ECC function during the read/write sequence [21], [35]. The whole test-case seems to be very similar to the paper proposal described in the previous sections:

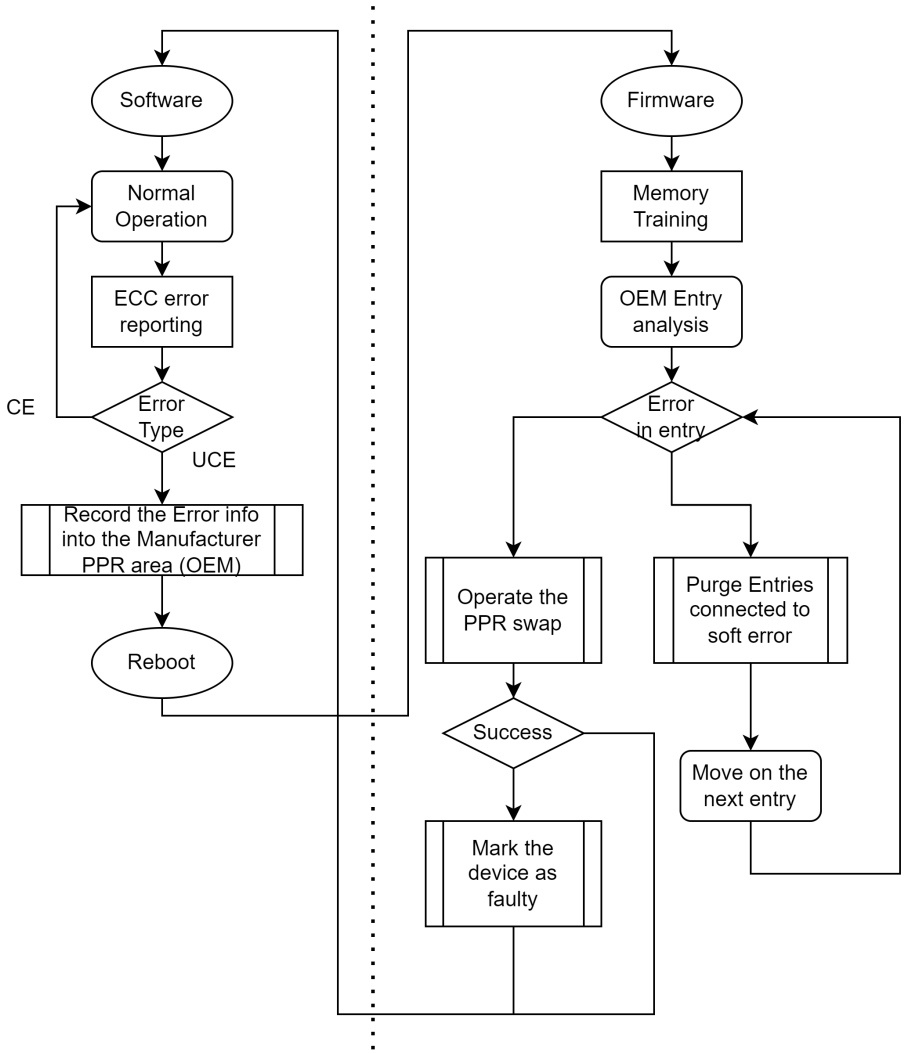

Fig. 7. The Schematic PPR flow char used in Production test

- ECC enabled at the start of the test;
- When ECC detects and communicates a UCE, the error information is stored in an area set up by the memory controller manufacturer (Original Equipment Manufacturer, OEM) to manage PPR entries.
- A restart of the board is performed;
- Following the restart, the firmware extracts the error information from the manufacturer area. It proceeds with the swap between fault and spare PPR entries if the memory location is still affected by an error after re-initializing the device.

As can be easily verified by comparing the two flows (see Fig. 5, Fig. 6 and Fig. 7), the mechanism we have defined for the run-time contemplates and therefore replaces the PPR implementation algorithm for the production test. With the implementation of the hard error management algorithm in run-time, we have uniformity in the memory error management mechanism in all the life phases of the product, realizing savings in development and maintenance. Since our proposal has no dependence on the manufacturer (thanks to using the inter-work data alternative to the OEM), it is a multi-vendors supporting solution.

VII. Conclusions and future work

The memory error handling proposed in this paper is particularly effective for network nodes that handle a significant amount of traffic with soft real-time requirements. Both the observance of the deadline and the supervision of a control system only allows a short time for the management of exceptions, such as those associated with uncorrectable error events in the system memory in the case of the use of PPR in the domain of exceptions. The proposed solution uses the processor restart to trigger the recovery action, which also provides for the isolation of row memory in case of a hard error and its replacement with a spare one via PPR. For a practical limitation of the number of restarts, we have used the ability to distinguish between data memory and control memory. In the first case, recovery action can

change the memory context without compromising the node's functioning, allowing a simple test operation to verify and remove the soft error.

However, the operating system cannot know how the applications will use memory, and this implies that effective use of this limitation on restart numbers must be able to rely on a different memory allocation API to allow the operating system to distinguish between data and control areas. How to provide and supervise data and control memory is undoubtedly research grounds for a future study. Future work can also look at integrating the control system to manage the degraded function state. It is the reason why the proposal suggested to add memory error information into the error log. Higher layers could use this information, such as the application or control system layer, to review the memory map and evaluate any degraded function states. Our study is limited to the integration between hardware, firmware, and software, while the management of resource availability limitation requires the involvement of the system layer.

REFERENCES

[1] S. Cha, O. Seongil, H. Shin, S. Hwang, K. Park, S. J. Jang, J. S. Choi, G. Y. Jin, Y. H. Son, H. Cho, J. H. Ahn, and N. S. Kim, "Defect analysis and cost-effective resilience architecture for future dram devices," *Proceedings - International Symposium on High-Performance Computer Architecture*, pp. 61–72, 5 2017.

[2] S. Khan and S. Hamdioui, "Trends and challenges of sram reliability in the nano-scale era," in *5th International Conference on Design & Technology of Integrated Systems in Nanoscale Era*, 2010, pp. 1–6.

[3] A. Yao, J. F. Li, F. Wang, J. Zhao, H. Liu, J. Zhang, J. Zhang, A. Zhou, Y. Song, J. Xu, P. Sun, K. Zhu, N. Ahuja, D. Zhu, and S. Kuo, "A memory ras system design and engineering practice in high temperature ambient data center," *InterSociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems, ITHERM*, vol. 2020-July, pp. 1379–1388, 7 2020.

[4] J. Technology, "DDR5 SDRAM," *JESD79-5B, JEDEC, Global Standards for Microelctronics Industry*, 8 2022.

[5] ——, "DDR4 SDRAM standard," *JESD79-4D, JEDEC, Global Standards for Microelctronics Industry*, 7 2021.

[6] C. Vitucci, D. Sundmark, M. Jägemar, J. Danielsson, A. Larsson, and T. Nolte, "Fault management framework and multi-layer recovery methodology for resilient system," *6th International Conference on System Reliability and Safety (ICSRS)*, pp. 1–8, 11 2022.

[7] R. W. Hamming, "Error detecting and error correcting codes," *The Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, 1950.

[8] V. Sankaranarayanan, "Ecc in ddr memories — designware ip — synopsys," *©2022 Synopsys, Inc*, 2022. [Online]. Available: https://www.synopsys.com/designware-ip/technical-bulletin/error-correction-code-ddr.html

[9] Y. Li, B. Nelson, and M. Wirthlin, "Reliability models for sec/ded memory with scrubbing in fpga-based designs," *IEEE Transactions on Nuclear Science*, vol. 60, no. 4, pp. 2720–2727, 2013.

[10] R. Rooney and N. Koyle, "Micron® ddr5 sdram: New features," *Micron Technology Inc., Tech. Rep*, 2019.

[11] C. Vitucci, D. Sundmark, M. Jägemar, J. Danielsson, A. Larsson, and T. Nolte, "A reliability-oriented faults taxonomy and a recovery-oriented methodological approach for systems resilience," *Proceeding IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*, pp. 48–55, 6 2022.

[12] R. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 305–316, 2005.

[13] L. H. Martínez, S. Khursheed, and S. M. Reddy, "Lfsr generation for high test coverage and low hardware overhead," *IET Computers and Digital Techniques*, vol. 14, 2020.

[14] S.-K. Lu, C.-L. Yang, Y.-C. Hsiao, and C.-W. Wu, "Efficient bisr techniques for embedded memories considering cluster faults," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 2, pp. 184–193, 2010.

[15] O. Wada, T. Namekawa, H. Ito, A. Nakayama, and S. Fujii, "Post-packaging auto repair techniques for fast row cycle embedded dram," 2004, pp. 1016–1023.

[16] J. Jeddeloh, "System and method for remapping defective memory locations," U.S. Patent 5 862 314, Jan 19, 1999.

[17] B. Querbach, R. Khanna, S. Puligundla, D. Blankenbeckler, J. Crop, and P. Y. Chiang, "Architecture of a reusable bist engine for detection and autocorrection of memory failures and for io debug, validation, link training, and power optimization on 14-nm soc," *IEEE Design & Test*, vol. 33, pp. 59–67, 2016.

[18] H. Lee, H. Oh, and S. Kang, "On-chip error detection reusing built-in self-repair for silicon debug," *IEEE Access*, vol. 9, 2021.

[19] S.-K. Lu, C.-J. Tsai, and M. Hashizume, "Integration of hard repair techniques with ecc for enhancing fabrication yield and reliability of embedded memories," 2015, pp. 49–54.

[20] R. Manasa, G. Hegde, and M. Vinodhini, "Improving the reliability of embedded memories using ecc and built-in self-repair techniques," 2018, pp. 1436–1439.

[21] D.-H. Kim and L. S. Milor, "Ecc-aspirin: An ecc-assisted post-package repair scheme for aging errors in drams," 2016, pp. 1–6.

[22] D.-H. Kim and L. Milor, "An ecc-assisted postpackage repair methodology in main memory systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, pp. 2045–2058, 2017.

[23] C.-L. Chao, S.-H. Wang, and H.-T. Wei, "Post package repair failure memory location reporting system," U.S. Patent 11 106 529B1, Aug 31, 2021.

[24] E. Pope L, "Row repair of corrected memory address," U.S. Patent 2019/0 019 569A1, Jan 17, 2019.

[25] J. Lee and K. Bains S., "Inline buffer for in-memory post package repair (ppr)," U.S. Patent 2020/015 070A1, May 14, 2020.

[26] S. Muthilayu, Y. Chen, Y. Yu, and T. Xu, "Runtime cell row replacement in a memory," U.S. Patent 2021/0 191 829A1, Jun 24, 2021.

[27] V. Zimmer, A. Agrawal, D. Wu, S. Ge, and Z. Wu, "Runtime post package repair for memory," U.S. Patent 2020/118 502A1, Jun 18, 2020.

[28] K. Tierney and M. Bruneau, "Conceptualizing and measuring resilience: A key to disaster loss reduction," *TR News*, 2007.

[29] A. Al-Dulaimi, X. Wang, and I. Chih-Lin, *5G networks: Fundamental requirements, enabling technologies, and operations management*, 2018.

[30] E. C. Strinati, M. Peeters, C. R. Neve, M. D. Gomony, A. Cathelin, M. R. Boldi, M. Ingels, A. Banerjee, P. Chevalier, B. Kozicki, and D. Belot, "The hardware foundation of 6g: The new-6g approach," 2022, pp. 423–428.

[31] *Common Public Radio Interface: eCPRI Interface Specification*. eCPRI Specification, V2.0, 2019.

[32] 3GPP, *38.801, Study on new radio access technology: Radio access architecture and interfaces*. Technical Report, version 14.0.0, Release 14, 2017.

[33] ——, *23.501, System architecture for the 5G System (5GS)*. Technical Specification, version 18.1.0, Release 18, 2023.

[34] S. Alnatheer and M. A. Ahmed, "Optimal method for test and repair memories using redundancy mechanism for soc," *Micromachines*, vol. 12, no. 7, 2021. [Online]. Available: https://www.mdpi.com/2072-666X/12/7/811

[35] Y.-S. Lee, J.-B. Lee, C. Kim, S. uhn Cha, and H. Yoon, "An efficient self post package repair algorithm and implementation in memory system with on-chip-egg," *2006 IEEE Asian Solid-State Circuits Conference*, pp. 331–334, 2006.