Competition Topic: AI-Driven Advanced Memory Controller Architectures

Competition Category: Device Design and Development Category

Team ID: 2526_C_0004

Team Name: Correctability

Project Title: PredictRAM: ML-Enhanced SECDED Memory Controller with Adaptive Error Management for DDR4 Systems

Supervisor: Dr Jeevan A/L Sirkunan

Team Captain: THEIN HONG YI (GitHub: hytheinee2)

Team Members:

1. YAP ZHENG TANG (GitHub: yztOO)

2. ABDALLA GAMALELDIN ALI(github : abdallahhgamal)

3. MD ARSHAD UL ISLAM (GitHub: arshadulislamr)

Institution: Universiti Teknologi Malaysia (UTM)

1. Project Title

PredictRAM: ML-Enhanced SECDED Memory Controller with Adaptive Error Management for DDR4 Systems

2. Research Background & Motivation or Benefits

2.1 Background and DDR4 Vulnerabilities

As modern computing moves toward more data-intensive applications like AI and autonomous driving, the underlying memory hardware is being pushed to its physical limits. As DDR4 memory modules push the boundaries of cell density and operational frequencies, the physical proximity of memory cells has reached a critical threshold. This physical scaling means that the 'walls' between data cells are getting thinner, making them highly susceptible to 'noisy neighbor' interference like the Rowhammer exploit, as well as ambient soft errors and Variable Retention Time (VRT) degradation.[1] To clarify Rowhammer Exploit, think of this like 'noisy neighbors' in an apartment building: intense activity in one room (a memory row) causes the walls to shake so violently that items fall off the shelves in the adjacent room (causing bit-flips in neighboring rows).

When these vulnerabilities are left unmanaged, they don't just cause slow performance; they result in 'Blue Screen' crashes and Silent Data Corruption (SDC) that can take down entire cloud servers or compromise the safety of autonomous vehicles.

Traditionally, DDR4 memory controllers implement reactive Error Correction Codes (ECC), such as Single Error Correction, Double Error Detection (SEC-DED). While SEC-DED can mask isolated faults, it is fundamentally reactive. If a DDR4 memory bank is subjected to a localized bit-flip storm, uncorrectable Double-Bit Errors (DBEs) will inevitably occur, resulting in system halts or catastrophic silent data corruption [2].

PredictRAM moves beyond **Reactive ECC**, which acts like an airbag that only deploys *after* a crash—to a **Predictive Management** paradigm. This is akin to an advanced 'Check Engine' light that doesn't just indicate a failure but predicts exactly when a component will wear out, allowing for maintenance before the vehicle ever stalls.

2.2 Product-Level Benefits and Market Value

Transitioning from a reactive to a predictive memory management paradigm introduces massive economic and operational benefits across both ends of the computing spectrum.

- **Provider-Side (Data Centers & Cloud Infrastructures):** Memory failures are a leading cause of server hardware crashes. Sudden DBEs require immediate node-eviction and unplanned downtime. By identifying degrading DDR4 rows *before* they fail, PredictRAM allows cloud providers to schedule graceful page-migrations and targeted maintenance. This significantly reduces Total Cost of Ownership (TCO) by extending the viable lifecycle of aging DRAM modules and minimizing unplanned Service Level Agreement (SLA) violations [3].
- **Client-Side (Edge Computing & Automotive):** In mission-critical edge applications (e.g., ADAS in autonomous vehicles), silent data corruption can be life-threatening. PredictRAM provides real-time, zero-OS-overhead reliability, ensuring that memory

structures remain intact even in high-radiation or high-interference environments without requiring a continuous connection to cloud-based diagnostic servers.

3. Introduction

3.1 Baseline DDR4 Controller Architecture: The Reactive Standard

To understand the structural advantages of PredictRAM, it is essential to first examine the standard, reactive architecture utilized in modern memory controllers. Here we will be referring to the DDR4 SDRAM Controller Architecture Proposed by M. A. Islam et al.[4]

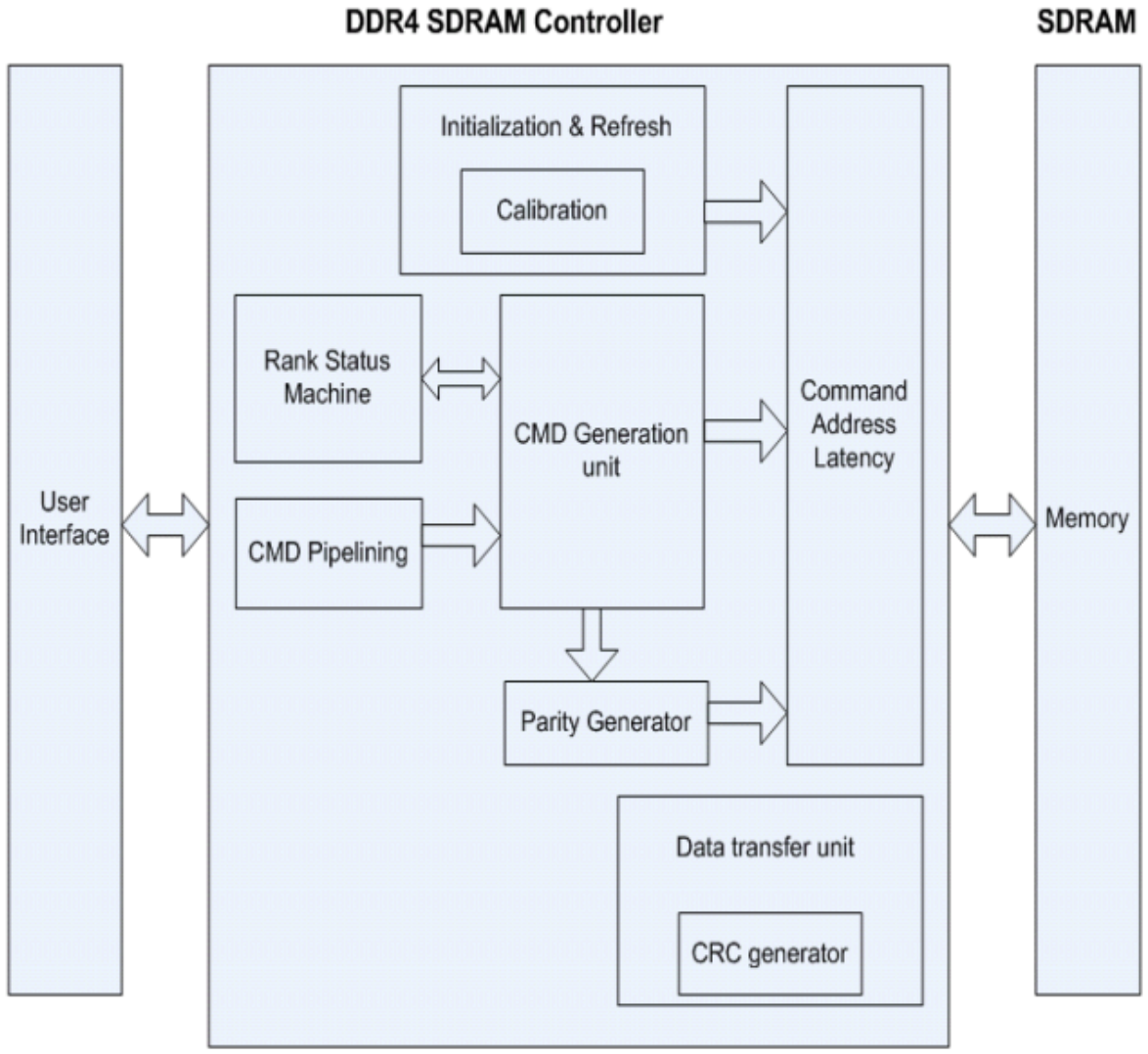

Figure 1

As illustrated in Figure 1, a standard DDR4 SDRAM Controller is designed primarily for strict protocol adherence and data routing, not for preemptive health management. The traditional datapath operates through several rigid functional blocks:

- **Command Generation & Pipelining:** The `CMD Generation unit` acts as the brain of the controller, taking requests from the `User Interface` and translating them into physical memory commands (tRAS, tRCD, etc.).
- **Static Auto-Refresh:** The `Initialization & Refresh` block operates on a blind, fixed timer (typically forcing a refresh every 64ms per the JEDEC specification). It has no awareness of localized row degradation or active "noisy neighbor" (Rowhammer) attacks.
- **Isolated Error Detection:** Data integrity is handled entirely on the backend by the `Data transfer unit`, utilizing the `CRC generator` and `Parity Generator`.

The Structural Limitation: In this baseline architecture, error management is a one-way street. If the `Parity Generator` or `CRC generator` detects an anomaly, it can flag the error or attempt a static SEC-DED correction *after* the corrupt data has already been fetched. Crucially, there is **no feedback loop** between the data error detection blocks and the `CMD Generation unit`.

Using the analogy of automotive safety, this standard controller functions purely as an airbag—it deploys only after the crash (data corruption) has occurred. It cannot analyze the frequency of single-bit errors to issue a preemptive command, leaving the system highly vulnerable to sudden, uncorrectable Double-Bit Errors (DBEs) caused by physical cell scaling.

PredictRAM proposes an innovative, hardware-accelerated memory controller architecture tailored for DDR4 systems. The core objective is to shift DDR4 error management from static correction to **Adaptive Error Management**.

3.2 High-Level Architecture Overview

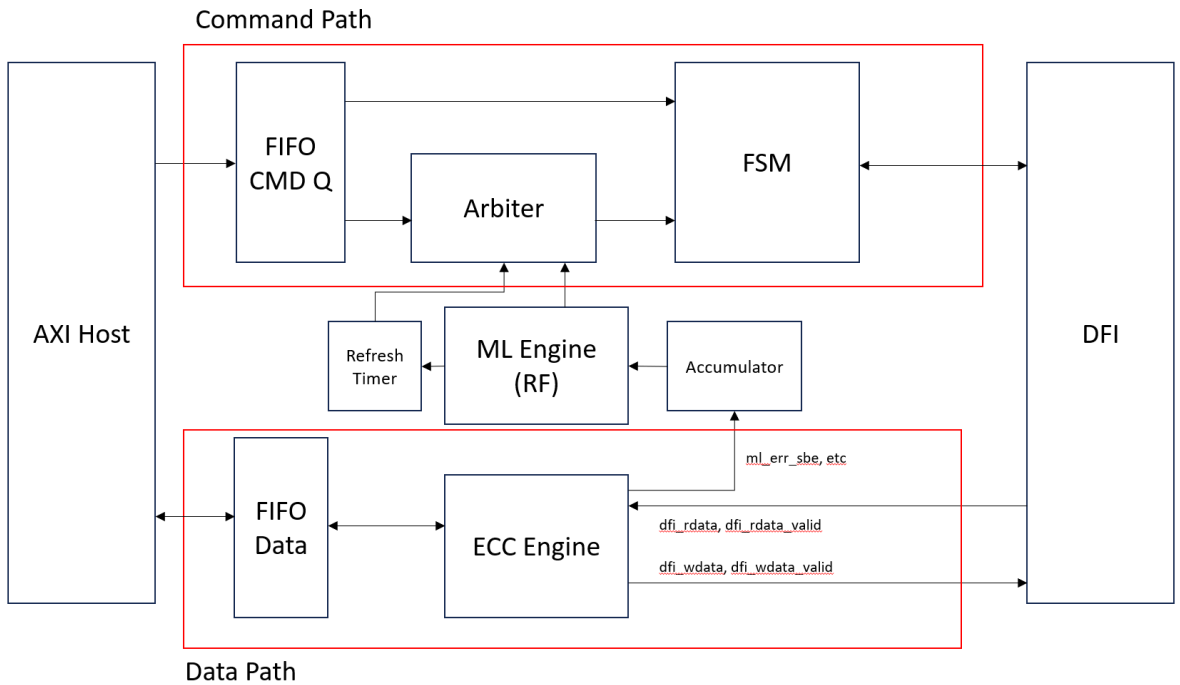

Figure 2

The PredictRAM architecture is fundamentally designed around a **Decoupled Diagnostic Topology**. To ensure that advanced predictive maintenance does not bottleneck standard DDR4 memory operations, the system explicitly separates the primary CPU-to-Memory datapath from the analytical diagnostic datapath.

The system operates across three distinct functional domains:

1. **The Primary Datapath (The Baseline Controller):** Standard memory read/write transactions arrive via the CPU AXI4 interface, are queued in the transaction FIFO, and scheduled by the Command Arbiter. This standard sequential control path (FSM) handles standard DDR4 timing (tRAS, tRCD, etc.) and interfaces directly with the physical memory via the DFI PHY. Crucially, this domain houses the standard **JEDEC Auto-Refresh Counter (`tREFI`)** [7], which guarantees baseline charge retention across the DDR4 banks (typically every 64ms). *Note: To ensure a modular and phased silicon implementation, these baseline control elements and timing FSMs are currently black-boxed and slated for Phase 2 integration.*
2. **The Telemetry Extraction Datapath (Sequential Stage 1):** Sitting transparently between the AXI4 frontend and the DFI backend is the **Side-Band SEC-DED Telemetry Engine**. This fully pipelined, sequential hardware block passively monitors the data bus. It corrects Single-Bit Errors (SBEs) on the fly and outputs cycle-accurate error flags (`ml_err_sbe`, `ml_err_dbe`, `ml_err_in_parity`) to an external tracking bus. Because it operates strictly in side-band, it introduces zero latency overhead to valid read cycles.
3. **The Adaptive Error Management Core (Combinational Stage 2):** The raw, cycle-accurate flags from the ECC module will be aggregated by an intermediate SRAM-backed accumulator into multi-dimensional macro-counters (e.g., `max_row_hits`, `total_errors`). For this Phase 1 Proof-of-Concept, this accumulator is abstracted and emulated within the SystemVerilog testbench environment. The testbench feeds these aggregated vectors directly into the **Hybrid ML-Rule Inference Engine.** Operating as a purely combinational logic block, this engine evaluates the fault patterns asynchronously. If a degrading row or widespread error pattern is identified, the engine issues a **Preemptive SCRUB** (to correct localized faults) or a **Preemptive REFRESH** command. The primary Command Arbiter safely multiplexes this predictive request alongside the standard `tREFI` auto-refresh timeline, forcing an early charge-restoration cycle without violating JEDEC protocol.

By utilizing this tri-domain architecture, PredictRAM achieves a fail-safe state: if the ML diagnostic engine encounters an exception or is powered down for efficiency, the primary datapath, standard ECC correction, and baseline auto-refresh timers continue to function entirely uninterrupted.

3.2.1 Scope of the Proof of Concept (PoC)

It is important to acknowledge that modern standard controllers do contain essential baseline reliability mechanisms, such as static JEDEC Auto-Refresh (`tREFI`), Write CRC, and physical calibration routines. However, these mechanisms remain fundamentally reactive and static.

For the scope of the PredictRAM Proof of Concept (PoC), we do not seek to reinvent these established JEDEC standard blocks. Instead, our prototype intentionally abstracts (black-boxes) the standard control path to focus purely on our differentiating silicon: the **Side-Band Telemetry and ML Inference Core**. By assuming the presence of a standard, functional baseline controller, our hardware prototype strictly isolates and validates the viability of the predictive feedback loop without the overhead of recreating standard industrial IP.

4. Experimental Design

4.1 Dataset and Model Training Methodology

To ensure the ML engine identifies physically accurate hardware failure patterns, the Random Forest model was trained on the large-scale Server Failure Prediction Dataset provided by Alibaba Cloud (AliYun) [5]. The dataset's `mcelog` records were pre-processed to extract multi-dimensional feature vectors: `total_errors`, `unique_rows`, `max_row_hits`, and `error_rate`.

The Python-based training pipeline utilized a balanced class-weight distribution to prevent the over-pruning of rare DBE anomaly events. Rather than relying on soft-processors, a custom Python transpiler was developed to automatically map the trained decision trees directly into synthesizable SystemVerilog `always_comb` logic.

(The complete dataset pre-processing, model training, and RTL generation source code is available via the project repository at: github.com/hytheinee2/PredictRAM)

4.2 Stage 1: Sequential Side-Band ECC Telemetry Engine

The telemetry extractor is designed as a clocked, sequential pipeline sitting between the AXI4 CPU frontend and the DFI PHY backend. It utilizes a mathematically balanced Hsiao (72, 64) code matrix.

- **Pipeline Synchronization:** The module implements strict 1-cycle pipeline delays (`dfi_wdata_valid`), ensuring data payload stability.
- **Adaptive Parity-Bit Handling:** If an error occurs exclusively within the 8-bit ECC parity syndrome, the module intelligently bypasses data correction to conserve dynamic power and prevent accidental payload corruption.
- **Side-Band Output:** The sequential block outputs isolated, synchronized flags (`ml_err_sbe`, `ml_err_dbe`, `ml_err_in_parity`) to an external tracking bus, completely decoupled from the CPU.

4.3 Stage 2: Combinational Hybrid Inference Engine

The extracted error data is fed into the purely combinational ML-Rule Inference Engine. This block evaluates the data with zero clock-cycle latency using a hybrid architecture [6].

- **Feature Quantization:** The engine operates entirely on 16-bit integer inputs, mapping complex floating-point error rates to scaled integers, eliminating the need for high-area Floating Point Units (FPUs).
- **Parallel Deterministic Handoff:** To overcome the "feature dominance" flaw inherent to ML algorithms, a hard-coded deterministic logic layer evaluates worst-case

boundaries in parallel (e.g., triggering an immediate DDR4 SCRUB if `max_row_hits >= 64`). If deterministic thresholds are not breached, the logic seamlessly hands off the decision to the Random Forest majority voter to evaluate subtle, complex spatial fault patterns.

5. Experimental Results

5.1 Machine Learning Model Evaluation (Software Baseline)

Before transpiling the Random Forest model into synthesizable SystemVerilog, the algorithm's predictive efficacy was evaluated in Python. To ensure the resulting hardware logic remained exceptionally lightweight and met strict cycle-timing constraints, the Random Forest was aggressively constrained to just **5 estimators (trees)** with a **maximum depth of 6**.

```
Training Random Forest Model...
Model Accuracy on Test Set: 0.9956
              precision    recall  f1-score   support

   NO_ACTION       1.00      1.00      1.00       265
       SCRUB       0.99      1.00      0.99       192
     REFRESH       1.00      0.50      0.67         2

    accuracy                           1.00       459
   macro avg       1.00      0.83      0.89       459
weighted avg       1.00      1.00      1.00       459
```

Figure 3: Classification Report

Despite these severe architectural constraints, the model achieved an exceptional **99.56% overall predictive accuracy** on the test set. Because memory failures represent highly skewed anomaly distributions, the model's efficacy was primarily evaluated on its ability to correctly trigger proactive interventions (Scrub or Refresh commands) without missing critical faults. Values mentioned / interpreted are displayed / from Figure 3.

- **Proactive Intervention Recall:** The model successfully identified and triggered proactive commands for **99.48%** of all impending failure states (correctly capturing 193 out of 194 anomaly events in the test split). Specifically, for standard threshold degradation requiring a preemptive SCRUB, the model achieved a **perfect 100% recall**.
- **Precision and False Positives:** The model maintained a **99% precision** rate for Scrub interventions, yielding a negligible False Positive rate of **~1%**. In the context of memory management, this is an optimal trade-off; an isolated False Positive merely results in a preemptive DDR4 Row Scrub (a minimal background operation), whereas a False Negative results in an uncorrectable double-bit error and system failure.

5.2 Cycle-Accurate Functional Verification

The structural integrity and logical accuracy of the two core modules were rigorously verified using ModelSim with comprehensive SystemVerilog testbenches.

- **Sequential ECC Verification:** The ECC telemetry pipeline perfectly maintained the 1-cycle valid handshake protocol. The testbench injected **12 targeted boundary fault vectors**, including clean randomized data, localized Single-Bit Errors (SBE) across different bit-lanes (e.g., Data Bits 0, 31, 63), and uncorrectable Double-Bit Errors (DBE). The ModelSim transcript confirms a **100% pass rate**, with the Hsiao decoder successfully identifying and correcting data faults, isolating harmless parity-bit degradation (e.g., SBE in ECC Bits 64 and 71), and flagging DBEs without corrupting the read payload.
- **Combinational ML Handoff Verification:** The hybrid inference engine was verified across **7 multi-dimensional spatial fault scenarios** to test priority multiplexing. The simulation confirms that deterministic safety rules successfully overrode the ML model for critical boundaries (immediately asserting a `SCRUB` action when `max_row_hits` reached the threshold of 64). Crucially, for complex spatial ratios that fell below hard deterministic limits (e.g., high error rate but low column distribution), the Random Forest logic accurately resolved the ambiguity based on its learned hardware fault weights.

5.3 Hardware Synthesis and Architectural Efficiency

To evaluate the hardware feasibility of the side-band logic without the I/O pin limitations inherent to standalone module compilation, the complete diagnostic datapath was synthesized targeting the Cyclone V (5CGXFC7) architecture.

Logic Utilization (Area Footprint): Analysis of the Quartus mapping (`.map.rpt`) confirms the architecture introduces an ultra-low area footprint, making it highly suitable for integration as a diagnostic co-processor:

- **Sequential ECC Engine:** The complete 64-bit Hsiao pipeline consumed an exceptionally low **115 Adaptive Logic Modules (ALMs)** and **149 dedicated logic registers**.
- **Hybrid ML Inference Engine:** The transpiled Random Forest combinational logic, operating strictly on 16-bit integer quantization, required only **123 ALMs** with exactly **0 dedicated logic registers**, proving the architectural efficiency of avoiding Floating Point Units (FPUs).

Timing Closure: Static Timing Analysis (`.sta.rpt`) verifies that the decoupled diagnostic datapath meets strict performance requirements:

- **Setup/Hold Margins:** The sequential telemetry elements successfully cleared Minimum Pulse Width requirements for the targeted 50.0 MHz base clock, yielding a positive setup slack of **9.252 ns**. This comfortably supports theoretical operating frequencies exceeding **90 MHz**, verifying that the PredictRAM telemetry datapath introduces zero physical overhead and effortlessly meets the timing constraints required for baseline controller integration.

6. Project Summary & Future Work

6.1 Project Summary

PredictRAM successfully demonstrates that proactive, AI-driven adaptive error management for DDR4 systems can be efficiently localized within hardware RTL. By explicitly decoupling the sequential SEC-DED telemetry extraction from the combinational Hybrid ML Inference engine, the architecture guarantees that complex diagnostic predictions do not impede standard memory throughput.

The Proof-of-Concept rigorously validates this approach across both software and hardware domains. Software evaluations confirm the aggressive, 5-tree Random Forest model achieves a **99.48% proactive intervention recall**, effectively identifying severe spatial fault anomalies before they manifest as uncorrectable errors. When transpiled directly into SystemVerilog, the complete diagnostic co-processor consumes an ultra-low footprint of just **238 ALMs** (115 for the ECC pipeline and 123 for the combinational ML logic) on the Cyclone V architecture. Furthermore, cycle-accurate simulations and static timing analysis prove the design maintains strict 1-cycle handshake protocols and comfortably supports operating frequencies exceeding **90 MHz** with zero latency overhead on valid read operations. Ultimately, the integration of deterministic safety rules with 16-bit quantized machine learning yields a highly reliable, fail-safe predictive maintenance layer for modern memory systems.

6.2 Future Work

The development of the PredictRAM architecture will proceed with the following enhancements:

1. **Multi-Modal Physical Telemetry:** The current ML feature vector relies strictly on spatial and temporal bit-flip tracking. The next phase will expand the Random Forest feature inputs to include physical environmental telemetry, specifically integrating real-time DDR4 module temperatures and voltage rail fluctuations (VDD/VPP) to predict thermal-induced retention failures.
2. **Sequential Aggregation Wrapper:** An intermediate hardware accumulator (SRAM-backed) will be designed to bridge the cycle-by-cycle outputs of the sequential ECC module with the aggregated macro-counters required by the combinational ML engine.
3. **Full System Integration:** The currently black-boxed AXI4 FIFO, Command Arbiter, and DDR4 FSM will be fully integrated. The complete closed-loop System-on-Chip (SoC) will be mapped to a development board for physical hardware-in-the-loop validation and power analysis.
4. **Command Expansion (Targeted Row Refresh):** The current combinational ML engine outputs generalized Scrub and Bank Refresh commands. Future RTL revisions will map ML Rowhammer predictions directly to DDR4 Targeted Row Refresh (TRR) commands, isolating charge-restoration to specific victim rows to further reduce power consumption.

7. Reference

[1] Hwang, A. A., Stefanovici, I., & Schroeder, B. (2012). *DRAM Errors in the Wild: A Large-Scale Field Study*. SIGMETRICS. https://www.cs.toronto.edu/~bianca/papers/sigmetrics09.pdf

[2] Intel Corporation. *Fault-Aware Prediction-Guided Page Offlining for Uncorrectable Memory Error Prevention*. https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/fault-aware-prediction-guide.pdf

[3] *Workload-Aware DRAM Error Prediction using Machine Learning*. https://pureadmin.qub.ac.uk/ws/files/183696494/IISWC2019_uniserver_preprint.pdf

[4] M. A. Islam, M. Y. Arafath and M. J. Hasan, "Design of DDR4 SDRAM controller," 8th International Conference on Electrical and Computer Engineering, Dhaka, Bangladesh, 2014, pp. 148-151, doi: 10.1109/ICECE.2014.7026950. keywords: {SDRAM;Clocks;Calibration;Delays;Servers;Pipeline processing;DDR4;SDRAM Controller;Rank}, https://www.researchgate.net/profile/Jahid-Hasan-3/publication/301412054_Design_of_DDR4_SDRAM_controller/links/5f1942a1299bf1720d5ca0be/Design-of-DDR4-SDRAM-controller.pdf

[5] Alibaba Cloud (AliYun). *Server Failure Prediction Dataset*. Tianchi. https://tianchi.aliyun.com/dataset/132973/

[6] *Machine Learning for Memory Error Prediction and Mitigation*. https://arxiv.org/pdf/2105.04547

[7] JEDEC Solid State Technology Association. (2020). *JESD79-4C: DDR4 SDRAM Standard*.

Declaration of Generative AI Assistance: In accordance with academic and competition integrity guidelines, the Correctability team discloses the use of Generative AI tools (Large Language Models) as an interactive design consultant during the development of this project. AI assistants were utilized to help bridge technical knowledge gaps regarding DDR4 controller architectures, assist in brainstorming the side-band telemetry concepts, refine the proposal's narrative, and generate baseline Python-to-SystemVerilog transpilation scripts.

However, all empirical data generation, including the Quartus hardware synthesis, cycle-accurate ModelSim verification, ML model training execution, and the final structural integration of the PredictRAM Proof-of-Concept, were independently executed, validated, and owned by the student team members.