GPU Silent Data Corruption Detection Based on Attention-Mechanism Metamorphic Relations

Xinyu Bai
xbai@illinois.edu

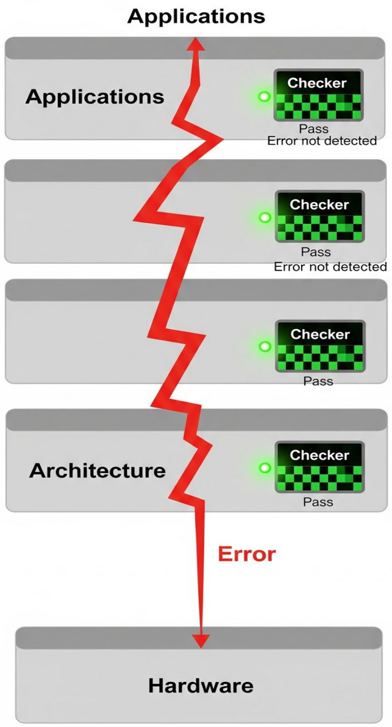
ILLINOIS

Table of Contents

Silent Data Corruption

- **Definition:** Incorrect outputs or state changes without any alert.

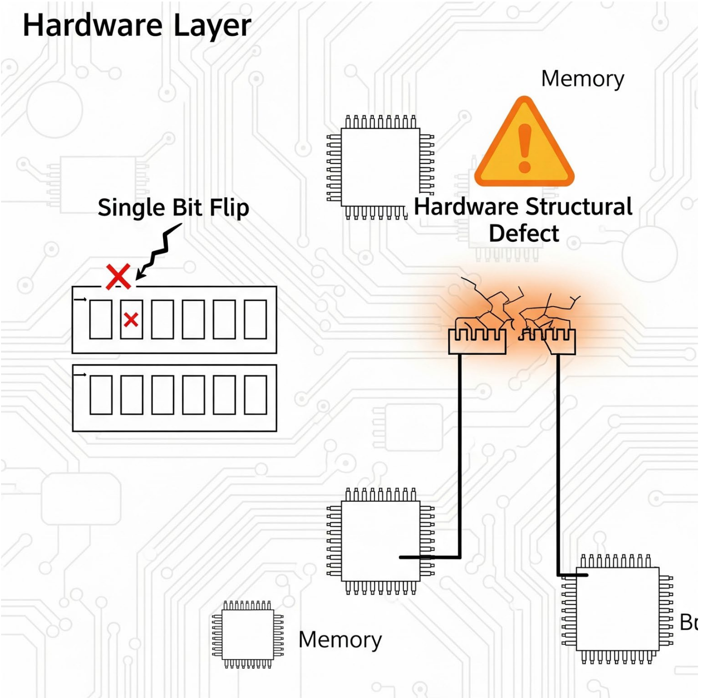
- **Status Quo:** In large-scale systems, many SDCs are repeatable and non-transient, but still lack effective methods to detect. [1, 2]



Generated by Gemini

Silent Data Corruption

- **Causes:**
  - Commonly attribute to transient single-bit flips caused by radiation.
  - Actually many originate from device or manufacturing process defects. [1, 3]

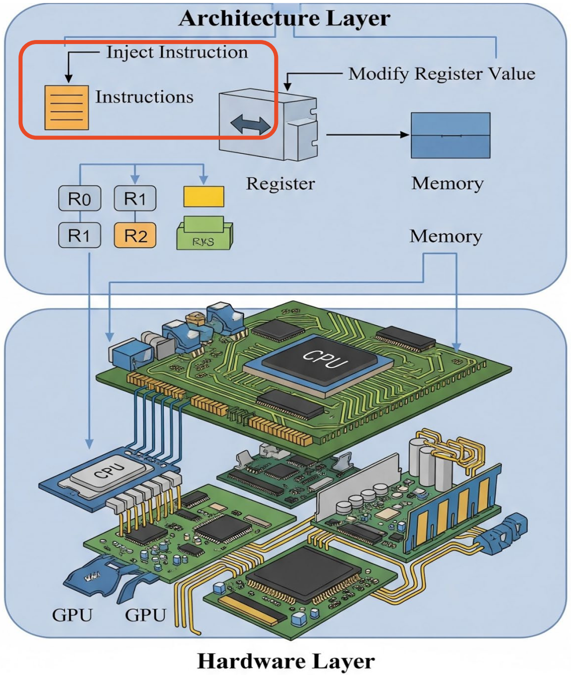


Generated by Gemini

Silent Data Corruption

- **Case 1: permanent structural defects**
  - Stable incorrcet results
  - Numerical anomalies
  - Control-flow abnormalities [1, 4, 5]

- **Simulators that model structure faults:**
  Instruction injection at the architectural level
  - **NVBitPERfi**
  - **HITPT**



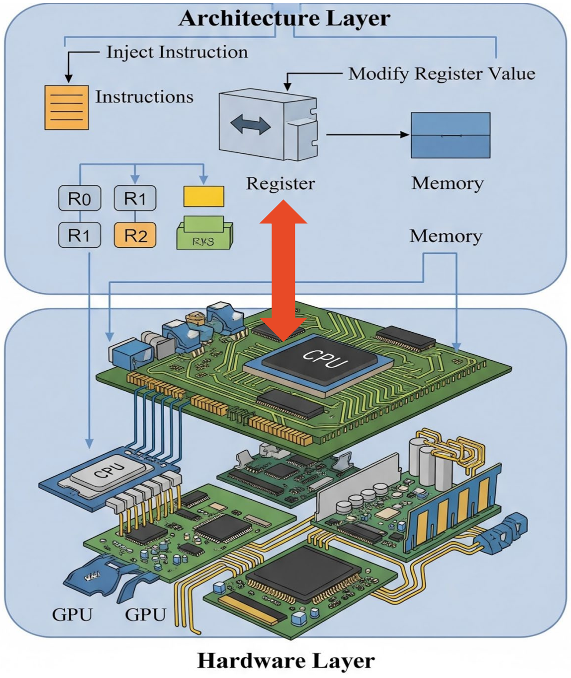
Generated by Gemini

Silent Data Corruption

- **Case 2: bit flip errors**
    - Most appears as visible single-bit flips at the architectural level, usually affecting 1 register and 1~2 threads. [6]

Inject hardware bit flips

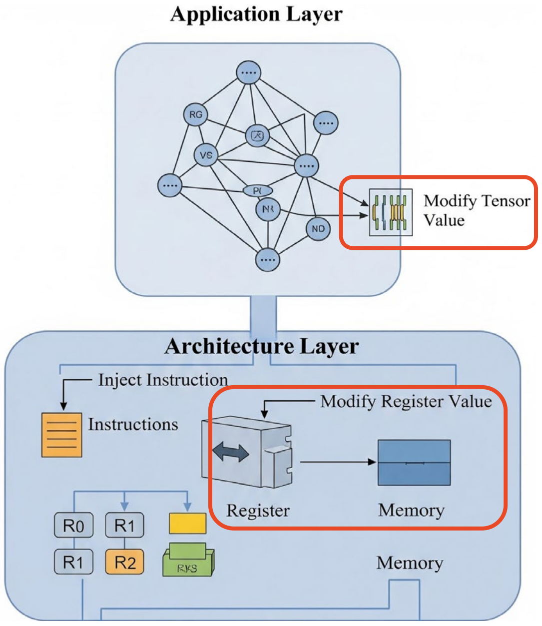
Inject bit flips in registers



Generated by Gemini

Silent Data Corruption

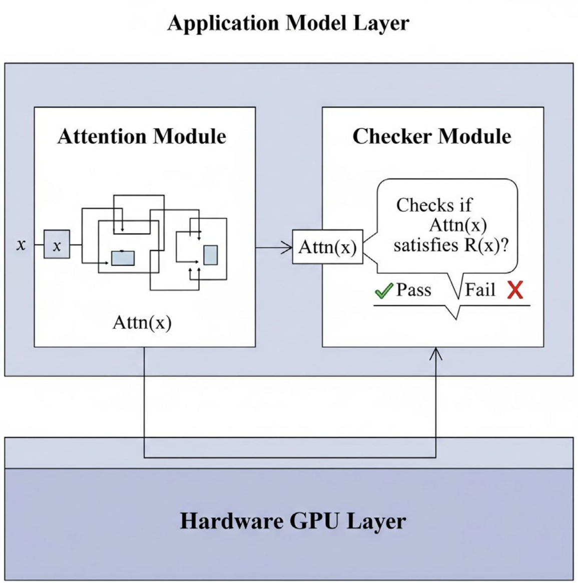
- **Single-bit flip in registers:**
  - 30%~35% result in segmentation faults
  - 65%~70% propagate to the application level
    - a small portion might be detected
    - the rest manifest as SDCs [4]
- **Simulators for bit flip:**
  - **NVBitFI:** registers at the architectural level
  - **PyTorchFI:** operators on the model side



Generated by Gemini

Our Job

- SDC mainly manifests as incorrect results:
  - Constructing metamorphic relations between model inputs and outputs
- Consider **attention module - bounds**
- Run simple injection experiments to evaluate feasibility



Generated by Gemini

Current Attention Bounds

Complexity Analysis

- Standard attention complexity:

$$\mathcal{O}(n^2 d)$$

- Theoretical bound verification:

$$\mathcal{O}(n \cdot (n + d^2))$$

Assume that $\forall\, \ell \in [n],\ k_\ell = v_\ell,$ and W is the **Lambert W function**

$$\text{Let } a_{ij} = \frac{q_i^\top k_j}{\sqrt{d}},\ w_{ij} = \frac{e^{a_{ij}}}{\sum_\ell e^{a_{i\ell}}},\ \mathrm{Attn}(x_i) = \sum_j w_{ij} v_j,$$

$$j^*_{(i)} = \arg\max_j a_{ij},\ \gamma_i = a_{j^*_{(i)}} - \max_{j \neq j^*_{(i)}} a_j,\ \varepsilon_i = q_i^\top k_{j^*_{(i)}} - q_i^\top \mathrm{Attn}(x_i),$$

$$\mathcal{W}_n = \mathrm{W}\left(\frac{n-1}{e}\right),\ \tau(\gamma_i, n) = \begin{cases} \dfrac{(n-1)e^{-\gamma_i}}{1 + (n-1)e^{-\gamma_i}}\gamma_i, & \gamma_i \geq \mathcal{W}_n + 1 \\ \mathcal{W}_n, & \gamma_i < \mathcal{W}_n + 1 \end{cases}$$

$$\text{Lower bound: } \varepsilon_i \geq \sqrt{d}\,\gamma_i\left(1 - w_{j^*_{(i)}}\right) \geq \sqrt{d}\,\frac{\gamma_i}{1 + e^{\gamma_i}}$$

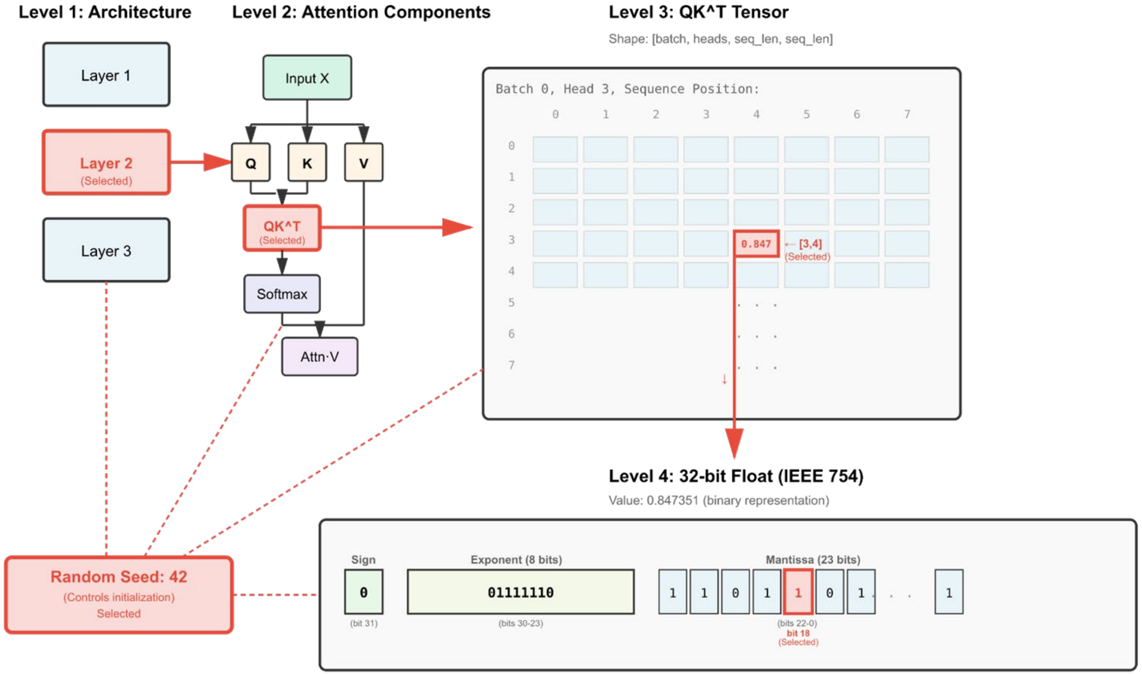
$$\text{Upper bound: } \varepsilon_i \leq \min\left\{ q_i^\top k_{j^*_{(i)}} - \frac{1}{n}\sum_j q_i^\top k_j,\quad \sqrt{d} \cdot \tau(\gamma_i, n) \right\}$$

Written in Typora, verified by GPT

Current Experiments

Start with simplest injection (single-bit flip in tensors):

- Random seed
- Attention layer
- Intermediate module
- Tensor index
- Bit position

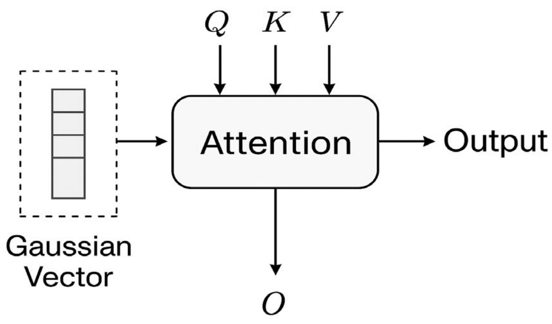


Generated by Claude, manually modified

Preliminary Experiments

Small scale experiment:

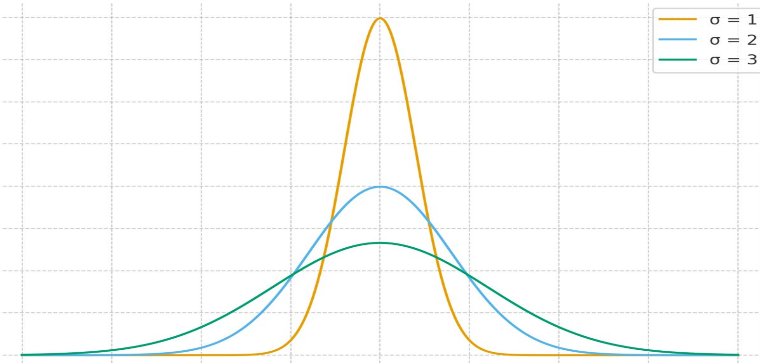
- Single layer attention
- Gaussian distributed random input



Generated by GPT

Results:

- Small range [-1, 1], hard to detect
- Expanded to [-3, 3], partially detectable, even for low order bit

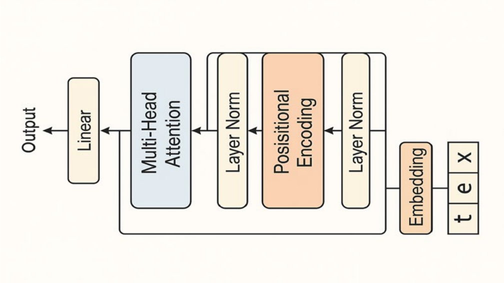


Generated by GPT

Preliminary Experiments

Extended experiment:

- Single layer Transformer (with Embed, FNN, Norm)
- Character-level dataset (one char as a token)



Generated by GPT

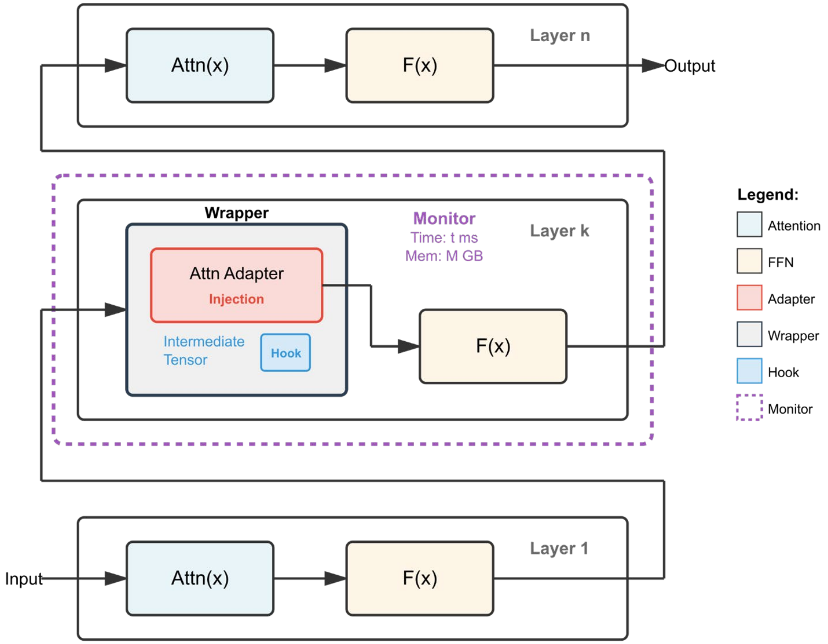
Results:

- Many NaN values
  - caused by **Mask Operation** in the implementation
- Still perform poorly after disabling mask
  - likely because vocab size (26) mapped to a 64-dim embedding, causing **high sparsity**

Current Experiments

Consider popular GPT-2 and WikiText

- Replace origin Attn with an adapter to inject
- An outer wrapper to return tensor copies
- A monitor to compare pre/post injection runtime, computation/ memory access overhead



Generated by Claude, manually modified

Current Experiments

Results: the detection is feasible

- injected bit position has the greatest impact, mostly concentrated in higher bits (≥23) - 30 >> 24 ≈ 25 > the rest bits
- Random seeds, injection layers, tensor indices remain roughly uniform
- K and Q, QK^T and Softmax are roughly pairwise balanced, ratio is approximately 3:2.

Current Experiments

Results

- Theoretical model requires K=V
  - Forcing K=V in the GPT-2 layer being injected
  - Rewriting the upper bound construction to depend on K

- Forcing K=V performs better than the rewritten upper-bound approach (Approximately 25% more detected)

Current Experiments

Results

- When forcing K=V, the tensor under test can be computed in two forms: Q * Attn(X) and QK^T * Softmax
- Both forms satisfy the upper and lower bounds, but each can detect different injection errors
- Taking the union of both detection results yields the best performance - 524 / 2048 = 25.6%

Future Directions

Bounds related:

- Extending the model to masked attention and further QKV modeling
- Develop bounds for backpropagation gradients
- Introduce special tokens / modules for testing
- For floating point number, construct tailored bounds
- Explore true metamorphic relations - x’ for a given token x, output remain the same

Future Directions

Experiment related:

- Port the method to fault simulators, or test on larger open source models
- Generalize the method to modern architectures (e.g., GQA, MLA)
- If tight bounds are difficult to obtain, consider an approximation-based approach - correction factors

References

[1] Dixit H. D. et al. Silent Data Corruptions at Scale. arXiv:2102.11245, 2021.

[2] Bonderson R. Silent Data Corruption in Systems at Scale. ITC Silicon Lifecycle Management Workshop, 2021.

[3] Wang S. et al. Understanding Silent Data Corruptions in a Large Production CPU Population. SOSP ’23 (ACM), 2023.

[4] Calhoun J. et al. Towards a More Complete Understanding of SDC Propagation. HPDC ’17 (ACM), 2017.

[5] Dixit H. D. Detecting Silent Data Corruptions in the Wild. arXiv:2203.08989, 2022.

[6] Hari S. K. S. et al. Estimating Silent Data Corruption Rates Using a Two-Level Model. arXiv:2005.01445, 2020.

[7] Vishwanathan M. et al. Silent Data Corruption (SDC) Vulnerability of GPU on Various GPGPU Workloads. Technical Report, Missouri S&T, 2016.

[8] Ma J. et al. Understanding Silent Data Corruption in LLM Training. arXiv:2502.12340, 2025.

Thank you for watching!