**Entropy-Aware Override Pathways in the ARCHON Core**

**Introduction**

This document defines the entropy-aware override pathways that govern instruction-level control within the ARCHON core. It formalizes the integration of quantum entropy and chaos signals, machine learning (ML) predictions, and traditional hazard detection into a multi-tiered pipeline override system. The goal is to create a robust and adaptive CPU control architecture capable of mitigating both deterministic and non-deterministic pipeline hazards.

**Entropy Signal Origins**

The ARCHON core incorporates several modules that generate signals indicative of system entropy, chaos, and higher-order anomalies. These signals are derived from various instruction-level inputs and internal CPU states, providing a holistic view of system stability and predictability.

* **quantum\_entropy\_detector**:
  + **Origin**: This module conceptually measures or simulates "quantum entropy" within the system.
  + **Instruction-level Inputs**:
    - instr\_opcode (from IF/ID stage): High entropy is associated with non-NOP, complex, or branching instructions.
    - alu\_result (from EX/MEM stage): Anomalous or unexpected ALU results (e.g., specific values when not expected) contribute to entropy.
    - zero\_flag (from EX/MEM stage): Unusual flag states in conjunction with alu\_result can indicate entropy.
  + **Output**: entropy\_score\_out (16-bit score): Increases with complex/anomalous instruction behavior and decreases during NOPs or stable operation.
* **chaos\_detector**:
  + **Origin**: Simulates a "chaos score" rising from unexpected or erratic system events.
  + **Instruction-level Inputs**:
    - branch\_mispredicted (from MEM/WB stage): Directly contributes to chaos, as mispredictions represent unexpected control flow.
    - mem\_access\_addr (from MEM stage): Erratic or unusual memory access patterns increase chaos.
    - data\_mem\_read\_data (from MEM stage): Unexpected data values read from memory at specific addresses can indicate chaos.
  + **Output**: chaos\_score\_out (16-bit score): Increases with mispredictions and erratic behavior, decays over time.
* **pattern\_detector**:
  + **Origin**: Identifies higher-order anomalous patterns within sequences of ALU flags.
  + **Instruction-level Inputs**:
    - zero\_flag\_current, negative\_flag\_current, carry\_flag\_current, overflow\_flag\_current (from EX stage): Tracks the history of these ALU flags over multiple cycles.
  + **Output**: anomaly\_detected\_out (1-bit flag): Asserts 1'b1 if any predefined anomalous pattern is detected across the flag history.

**Entropy–FSM Linkage**

The probabilistic\_hazard\_fsm module acts as the central orchestrator for pipeline control. It synthesizes inputs from the Archon Hazard Override Unit (AHO) and directly from an external ML model to determine the appropriate pipeline control signal (Normal, Stall, Flush, Lock).

* **archon\_hazard\_override\_unit (AHO)**:
  + **Inputs**: Integrates entropy\_score\_out, chaos\_score\_out, anomaly\_detected\_out, branch\_miss\_rate\_tracker, cache\_miss\_rate\_tracker, and exec\_pressure\_tracker.
  + **Dynamic Weighting**: The ml\_predicted\_action input from the external ML model dynamically adjusts the weights (Wentropy​, Wchaos​, etc.) applied to each metric. This allows the AHO to adapt its sensitivity and risk posture based on ML context. For example, a "Critical Risk" ML prediction will significantly amplify the impact of entropy and chaos scores.
  + **Combined Hazard Score**: All weighted scores are summed to form total\_combined\_hazard\_score.
  + **Threshold-Based Overrides**: This combined score is compared against dynamically scaled\_flush\_threshold and scaled\_stall\_threshold.
  + **Priority Rules**: anomaly\_detected\_out takes the highest priority; if an anomaly is detected, a flush is requested regardless of the combined score. Otherwise, the combined score determines if a flush or stall request is generated.
  + **Output to FSM**: aho\_override\_flush\_req and aho\_override\_stall\_req are generated. These are consolidated into fsm\_internal\_hazard\_flag (ORing both) for input to the FSM.
* **probabilistic\_hazard\_fsm**:
  + **States**: STATE\_OK, STATE\_STALL, STATE\_FLUSH, STATE\_LOCK.
  + **Inputs**: ml\_predicted\_action (direct ML override suggestion) and fsm\_internal\_hazard\_flag (from AHO).
  + **Override Tier Escalation**:
    - **ML Directives**: The ml\_predicted\_action can directly force the FSM into STALL, FLUSH, or LOCK states, overriding the AHO's input if the ML model deems it necessary.
    - **AHO Influence**: If the ml\_predicted\_action is OK (or implies no direct override), the fsm\_internal\_hazard\_flag (derived from AHO's flush/stall requests) can transition the FSM from STATE\_OK to STATE\_STALL.
    - **Escalation Path**: From STATE\_OK, if AHO detects a hazard, it can move to STALL. From STALL, ML can escalate to FLUSH or LOCK. From FLUSH, ML can escalate to LOCK.
    - **LOCK State**: This is a critical override tier. Once in STATE\_LOCK (triggered by an ml\_predicted\_action of 2'b11 or AHO's anomaly\_detected\_val triggering a flush and FSM deciding to lock), the FSM remains in LOCK until an external hard reset (rst\_n). This signifies a severe, unrecoverable system anomaly or attack.

**Override Pathway Table**

The following table summarizes how combinations of Entropy/Chaos, ML predictions, and internal hazards (from AHO) result in specific FSM overrides.

|  |  |  |  |
| --- | --- | --- | --- |
| Entropy/Chaos Score (AHO Input) | ML Predicted Action | Internal Hazard Flag (from AHO) | FSM Override Tier |
| Low | OK (2'b00) | Low/None | OK |
| Low | OK (2'b00) | High (e.g., AHO requests STALL) | STALL |
| Any | STALL (2'b01) | Any | STALL |
| High | OK (2'b00) | Low/None | STALL (AHO may request STALL based on high weighted scores) |
| High | OK (2'b00) | High (e.g., AHO requests FLUSH) | FLUSH (AHO may request FLUSH based on high weighted scores) |
| Any | FLUSH (2'b10) | Any | FLUSH |
| Any (especially anomaly\_detected\_val) | OVERRIDE/LOCK (2'b11) | Any | LOCK |

**Conclusion and Future Extension**

This formalized architecture provides a robust framework for adaptive CPU pipeline control by deeply integrating entropy-based metrics and ML-driven directives. The multi-tiered override system, culminating in a critical LOCK state, ensures comprehensive hazard mitigation.

**Future Extensions**:

* **Quantum Entropy Systems Integration**: The quantum\_entropy\_detector module is currently a placeholder. Future work could involve direct interfacing with a true quantum entropy source or a more sophisticated quantum state simulation, allowing the ARCHON core to dynamically adapt to quantum fluctuations.
* **QPU Scheduler Porting**: The principles of entropy-aware, ML-modulated hazard detection and multi-tiered overrides could be directly ported to Quantum Processing Unit (QPU) schedulers. Given the extreme sensitivity of quantum coherence to environmental noise (a form of entropy), dynamically adjusting QPU execution parameters (e.g., qubit refresh rates, gate execution order) based on real-time entropy and ML-predicted error rates would significantly enhance quantum program reliability and performance. This would involve mapping "pipeline stalls/flushes" to QPU-specific control actions like re-initialization, error correction cycles, or dynamic circuit re-compilation.