RTL Fault Model for RISC-V Control Subsyst

AUTHOR: G.PRASANTH

FAULT THEORY → RTL RULES

A structured approach to defining control-level faults in modular RISC-V subsystems before implementing detection logic.

CONTEXT

Project Context

Modular RISC-V Control Subsystem

Building a clean, verifiable control path designed with correctness and safety as primary objectives. The architecture prioritizes structured fault tolerance through early definition and disciplined design.

This project establishes fault models before implementing detection or recovery mechanisms, ensuring a solid theoretical foundation.

01

Define Fault Models

Establish theoretical framework

02

Implement Detection

Add monitoring logic

03

Enable Recovery

Future fault tolerance

MOTIVATION

Why Fault Modeling at RTL?

Earliest Detection Stage

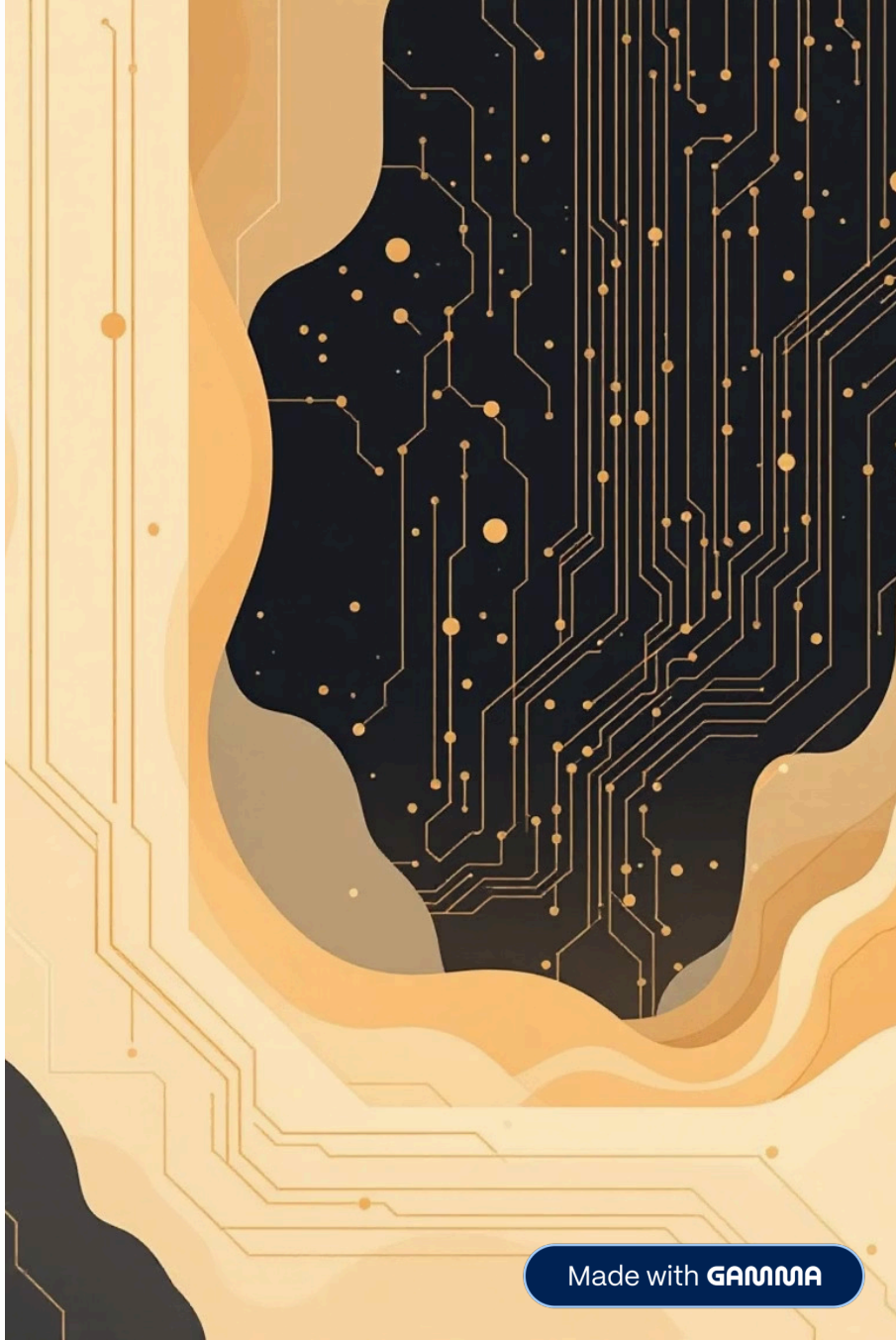
RTL is the first architectural level where unsafe control behaviors can be formally identified and prevented before silicon implementation.

Control Path Integrity

Logical control faults cause undefined execution states that can cascade into system-level failures or security vulnerabilities.

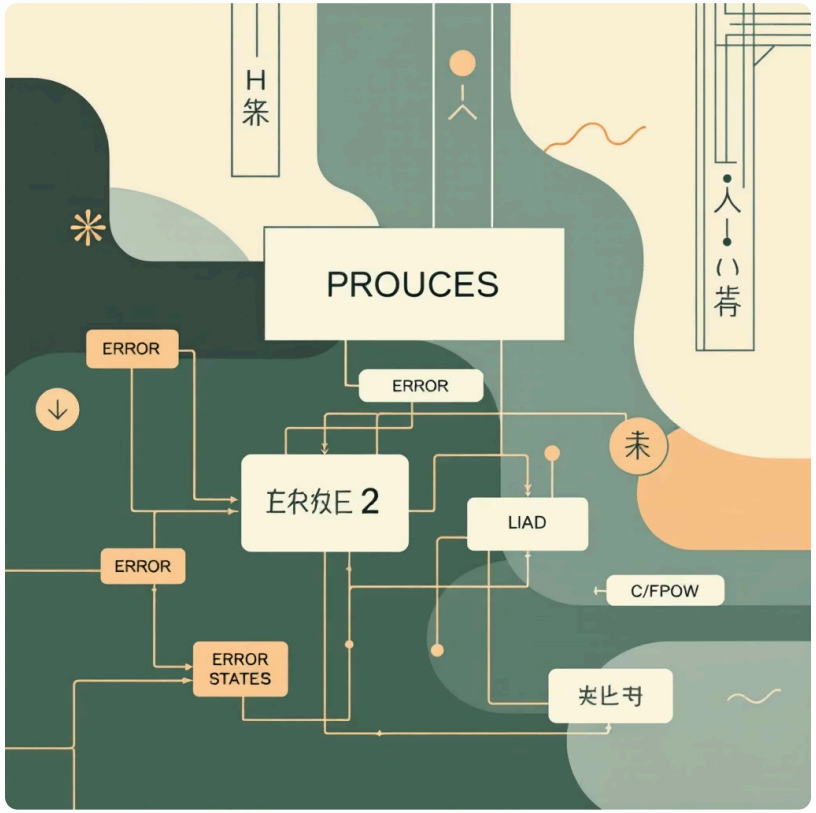
Simplified Recovery

Early fault definition at the design stage enables cleaner recovery architectures and reduces verification complexity.



DEFINITION

What Is a Fault at RTL Level?



> Any unsafe or undefined control behavior that violates design correctness, regardless of physical hardware state.

Key Insight: RTL faults focus on logical correctness and forward progress of the control subsystem. They represent violations of architectural invariants rather than physical defects.

Faults can occur even in perfectly manufactured hardware due to design oversights, unexpected state transitions, or incomplete specification coverage.

CLASSIFICATION

Fault Classification Overview

1

Invalid Opcode

Unsupported instruction in implemented subset

2

Illegal Control Combination

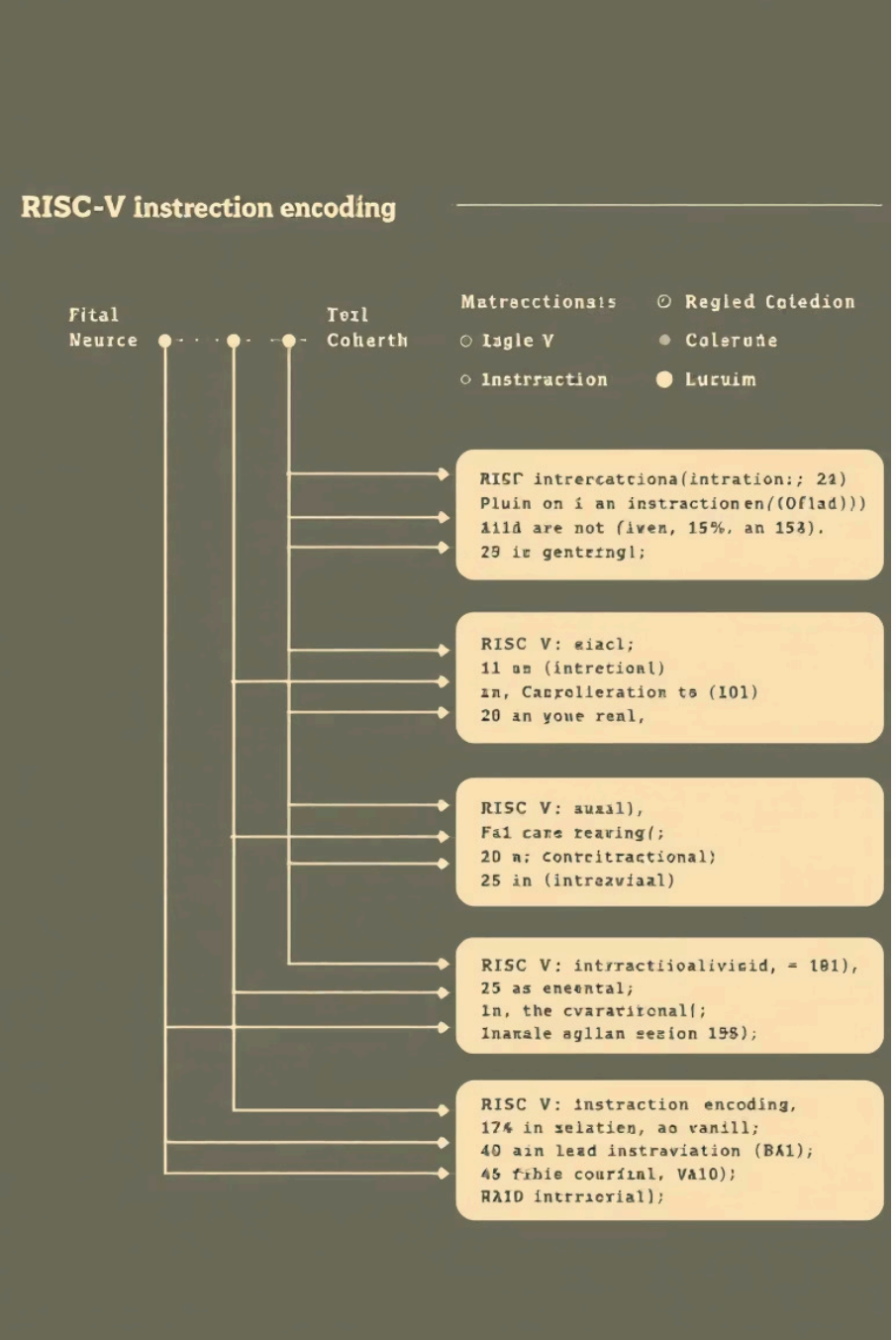
Mutually exclusive signals asserted simultaneously

3

Stuck-At Behavior

Control signal frozen across multiple cycles

Design-Level Focus: All fault types are independent of fabrication defects and represent architectural violations at the RTL abstraction level.



1 FAULT TYPE 1

Invalid Opcode

Definition

An opcode that does not exist in the implemented RISC-V instruction subset. The control decoder cannot safely generate control signals for execution.

RTL Rule

if (opcode ∉ SUPPORTED_SET)
→ FAULT_INVALID_OPCODE

Control signals remain in safe default state

Critical implication: Any unsupported opcode immediately triggers a fault condition, preventing undefined control signal generation.

FAULT TYPE 2

Illegal Control Combination

1

Detection

Monitor for mutually exclusive signals asserted together

2

Examples

- mem_read & mem_write both HIGH
- Multiple ALU operations active
- Conflicting branch conditions

3

Impact

Represents unsafe hardware behavior with unpredictable outcomes

---

Memory Conflict

`mem_read ∧ mem_write`

ALU Conflict

`alu_add ∧ alu_sub`

State Conflict

`branch_taken ∧ branch_not_taken`

FAULT TYPE 3

Stuck-At Behavior

Behavior Pattern

A control signal remains fixed at logic 0 or logic 1 across multiple clock cycles when it should be toggling based on instruction flow.

Detection window: Typically monitored over 2-4 cycle threshold to distinguish from valid static states.

Critical Examples

- `pc_en` stuck LOW → no forward progress
- `reg_write` stuck HIGH → register corruption
- `state[1:0]` stuck → FSM deadlock

`Impact:` Breaks instruction execution forward progress and violates fundamental architectural assumptions.

RULES

RTL Rules Summary

Single Valid Path

Exactly one valid control path must be selected per instruction decode cycle

Flow Adherence

Control signals must strictly follow the architectural instruction flow without deviation

Opcode Coverage

Any opcode outside the implemented subset is considered illegal and triggers immediate fault

Design Invariant: These rules form the theoretical foundation for all fault detection logic. Violations indicate control subsystem integrity failure.

Engineering Value & Next Steps

Current Scope: Definition Only

- No fault correction mechanisms
- No recovery logic implementation
- Pure fault model specification

Engineering Benefits

- Improves design robustness and discipline
- Enables structured fault detection architecture
- Provides verification framework foundation
- Reduces integration debugging time

Implementation Roadmap

Completed
Fault model defined

1

Next
Add fault flags

2

Future
Detection & recovery

3

THANK YOU