Multicore Avionics Operational Analysis

Lightning Talk 4: Design Part 1

Team: sddec24-09

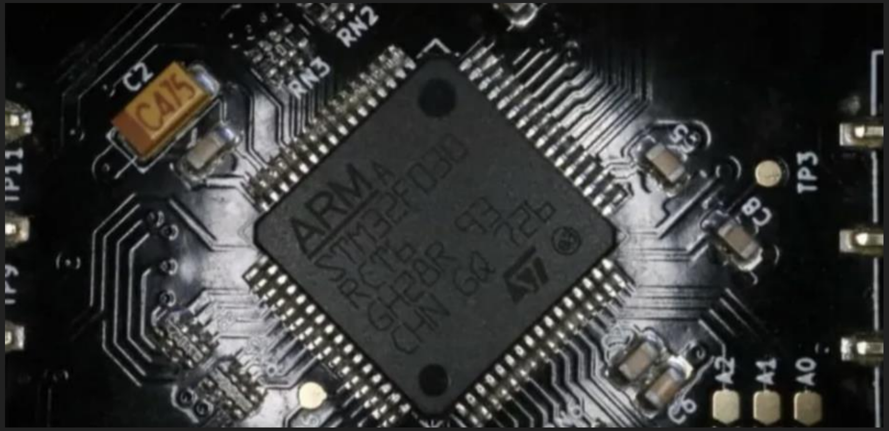
Team members: Alex Bashara, Joe Dicklin, Hankel Haldin, Anthony Manschula

Faculty advisors: Dr. Zambreno & Dr. Jones

Client: Boeing

Project Overview

- Multicore avionics systems
  - Meet the increasing compute demand of modern avionics software with concurrent execution of programs
  - Concurrent programs competing for shared resources
    - Introduce interference & negatively affect execution timing behavior
    - Ability to examine and verify the effects of interference is critical for FAA certification
- Hardware: ARM-based SBC and bare-metal hypervisor
  - Hypervisor allows more granular control of resource allocation to programs
  - Run control applications with the system under extreme load
    - Collect and analyze data on worst-case execution time (WCET)

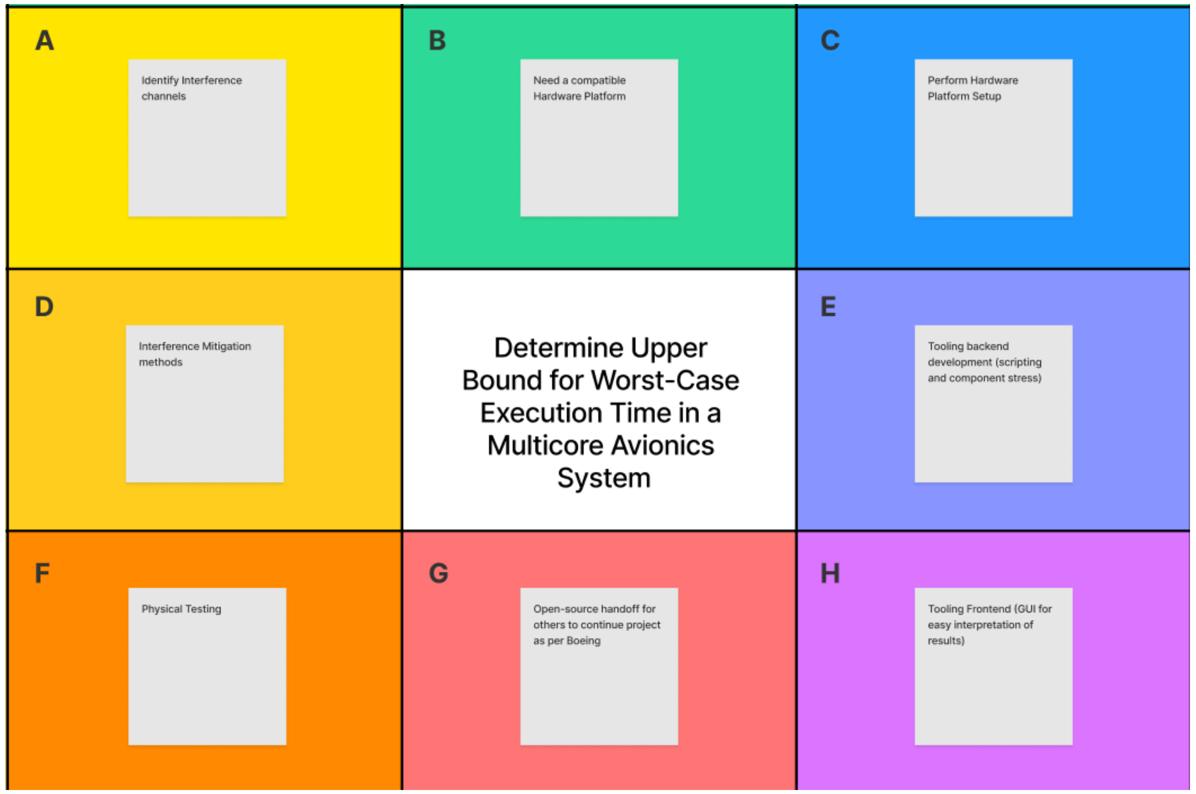


Problem Statement

- Our project addresses a need for a suite of open-source tools to characterize interference modes in multicore avionics systems
  - Identify potential interference channels on a multicore platform
    - "Control tests" as a baseline that target each channel for use in analysis
  - Set of tools to apply stress and contention to the identified subsystems in a controlled manner
  - Set of tools/methods to demonstrate mitigation of interference channels
  - Integrate testing and analysis tools into unified suite

Ideation

- Primary source of ideas: Lotus Blossom Diagram
  - Allowed us to explore different aspects or branches of our project could take
  - e.g, What hardware platform, existing interference generators, type of hypervisor will we use?
- Ultimately what guides our final design is input from our client
  - This is how we chose our hardware platform and supporting software tools


A
Identify Interference channels
B
Need a compatible Hardware Platform
C
Perform Hardware Platform Setup
D
Interference Mitigation methods
Determine Upper Bound for Worst-Case Execution Time in a Multicore Avionics System
E
Tooling backend development (scripting and component stress)
F
Physical Testing
G
Open-source handoff for others to continue project as per Boeing
H
Tooling Frontend (GUI for easy interpretation of results)

Solution

- It is difficult to describe one concrete solution for our project due to the complex nature of computer hardware, so we will describe an approach for each contention point
  - Processor cache: Rapidly dirty the cache, such that lines are repeatedly ejected and new ones fetched from memory
  - Memory bandwidth: Generate a large amount of access traffic to main memory
    - This can be done by a similar approach as above, since cache thrashing inherently needs to access memory
    - Disable caching for a portion of the page table in the OS and perform accesses to that region
  - I/O: Network-based, such as UDP packets at a bandwidth over what the connection can handle to induce stress on the subsystem

Market Research

- This project would be of interest to parties needing a method to verify an ARM-based system against DO-178 guidelines for a safety-critical or high-reliability application. Some potential users of this project could include:
  - Players in the aerospace industry such as Boeing, Lockheed Martin, Bombardier, etc.
  - High-reliability applications such as industrial automation or machine vision
    - Automakers incorporating full-self-driv ing technology like Tesla
    - Manufacturing defect detection, computer-vision assisted machining

For the Audience

- Conclusion
  - We are developing a solution guided by our client's input and our own research. The complex nature of our project requires us to quickly iterate on our ideas and adapt the ones that work. Ultimately, we hope to produce an open-source library that addresses industry needs.
- Questions?