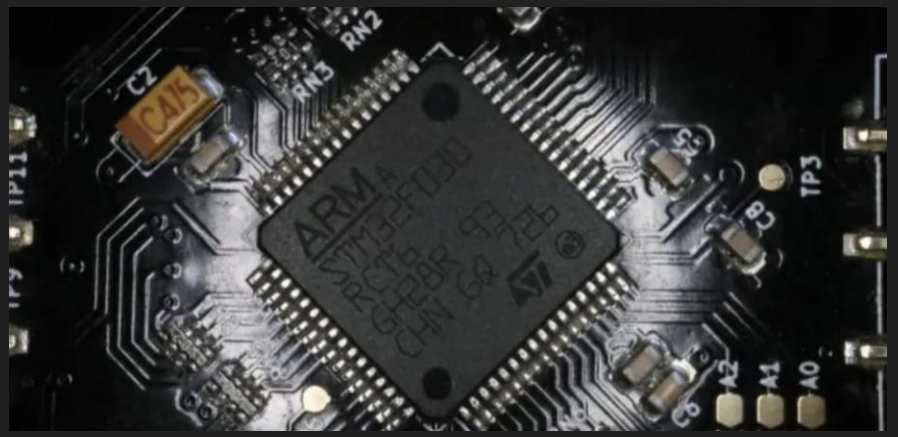
Multicore Avionics Operational Analysis

Lightning Talk 3: Project Plan

Alex Bashara, Joe Dicklin, Hank Haldin, Anthony Manschula

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

Project Management Style

- Hybrid
  - Waterfall-style task decomposition
    - e.g, a full two semester plan that decomposes tasks into research, implementation, testing, & presentation
  - Agile methods
    - Weekly meeting and status updates with our client
    - Track open action items and backlog with GitLab
    - Our client can monitor our GitLab backlog and actively guide its direction
- Waterfall gives our project structure on a longer timescale
- Agile allows us to adapt and handle more granular project requirements as they arise

Task Decomposition

- Hardware Bring up
  - Set up Xen hypervisor build environment with Yocto
  - Verify Xen for functionality as built
- Develop Base Test Cases
  - Base cases will perform core/memory/cache/IO-intensive workload
  - Baseline serving to establish a comparison metric
- Introduce Resource Contention/System Stress
  - Simulate "rogue" program affecting our base case
    - Observe the effect on execution time
- Mitigate Resource Contention
  - How can we reduce the impact of system stress on the base case?
- Unify Tests and System Stressors into Comprehensive Suite
  - Improve usability and presentation of the tool set

Metrics and Evaluation Criteria

- How do we know that we are actually doing something useful?
  - Primary metric: Base case execution time
    - For each tested subsystem (core/cache/memory, etc.), we should already know the impact of resource contention on the base case
      - Execution time increases, stays the same, etc.
      - Requires background knowledge of our platform's underlying hardware architecture
    - Perform many test runs with resource contention
      - Apply statistical analysis to confidently determine a worst-case execution time
  - Might also be useful to track system resource usage for debugging and validation purposes

Key Risks

- Risk: There is significant set up required to get to the core of our project
  - Hardware selection
  - Software installation & configuration
  - Research into existing tools & multi-core interference generators
- Risk: Acquiring hardware can be time consuming
  - Our project requires a specific ARM based SoC
  - No guarantee we can install & configure the necessary software on a particular platform
    - e.g, Raspberry Pi while ARM based, did not work for our project

Risk Consequences and Mitigation

- Consequence: our project timeline could be thrown off track
  - There is a lot of work to do across two semesters
  - If set-up takes too long it will leave less time for the overall goal of our design
- Mitigation: our client provides us with technical support to get us unstuck if necessary
- Mitigation: we can emulate interference on Linux VMs in an x86 environment
  - This allows us to tackle the core aspects of our design irrespective of hardware issues

For the Audience

- Conclusion
  - The team has many requirements to fulfill, but our choice of ISA, need for thorough and accessible documentation, as well as definitive proof of WCET are most critical.
- Questions?