IOWA STATE UNIVERSITY

Multicore Operational Analysis Tooling (MOAT)

Senior Design Team: sddec24-09
Client: The Boeing Company
Advisor: Joseph Zambreno

Alexander Bashara, Joseph Dicklin, Hankel Haldin, Anthony Manschula

Introduction

The increasing computational demand of modern avionics programs necessitates higher performance hardware platforms to support them. One approach to achieving higher application performance is to utilize a multicore system. However, incorporating such systems into safety-critical applications like avionics presents a unique set of challenges when it comes to their airworthiness certification. The equipment manufacturer must be able to prove that the system is resilient to performance degradation due to shared resource "crosstalk" (also known as *resource contention*) from applications running on the platform's processor cores. Our team was tasked with building a test framework that would induce sufficient resource contention on a target piece of hardware to facilitate more efficient airworthiness testing of embedded Linux systems.

Context

Users:

- Embedded Linux Hardware and Software Developers in Aerospace and Other Fields Requiring Safety-Critical Multicore Systems

Use Cases:

- Avionics Hardware/Software Development

Design Approach

Our final design is composed of the target hardware platform, hypervisor, and a command line interface. The hypervisor, sitting between the command line interface and hardware, is responsible for partitioning the underlying hardware resources of our platform by assigning them to separate virtual machines called Dom0's and DomU's. The user quantifies resource contention by running different configurations of base programs in Dom0 and interference generators in DomU's. Once an interference test completes, execution metrics are stored in a YAML file. The data file is exported for statistical analysis to determine Worst Case Execution Time.

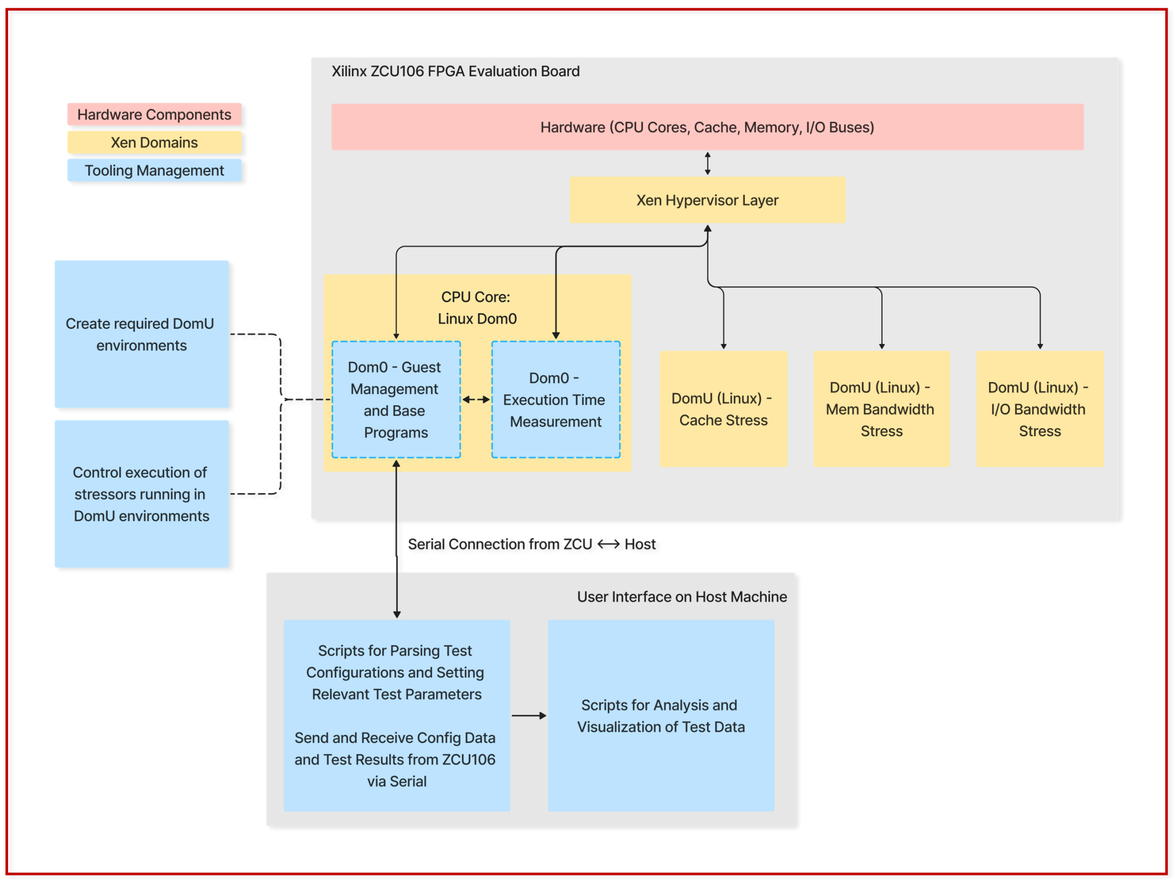

Figure 1: System Architecture

Design Requirements

- **Physical/Resource Requirements**
  - *ARMv8 processor subsystem*
  - *Form-factor: Single-board computer or FPGA board implementing a Xilinx MPSoC*
- **General User Knowledge Requirements**
  - *Linux environments*
  - *Worst-case execution time (WCET) and its influencing factors*
  - *Familiarity with multicore computer architectures*
- **Functional/Technical Requirements**
  - *Properly and methodically stress the system*
    - Ensure that the stress applied to the system is as intense as possible
  - *Identify major points of resource contention*
  - *Identify a lower bound on WCET*
  - *Provide an effective way of measuring and analyzing performance metrics of various runs, both with and without the system under stress, to estimate a worst-case execution time in those conditions*
- **UI (User Interface) Requirements**
  - *Develop a well-documented command line tool to work with our platform*
  - *Provide a user-friendly tool for managing and interpreting test results*

Implementation Details

Target Hardware Platform and OS

- Xilinx ZCU106 FPGA Development Board
  - 4x ARM Cortex A53 CPU Cores
  - Runs Custom PetaLinux 2023.1

Hypervisor

- Xen – Bare Metal/Type-1 Hypervisor
- Domains
  - Dom0 (Core 0) – PetaLinux 2023.1 – Collects Performance Metrics
  - DomU (Cores 1-3) – Ubuntu "Base" – Generates Resource Contention

Test Management and Tooling

- Test Harness – Parses User Configuration Files for Tests to Run
  - Python/YAML, Communicates via Serial
- Results Analysis – Parses Results, Perform Statistical Analysis & Visuals Generation
  - Python/YAML/Matplotlib

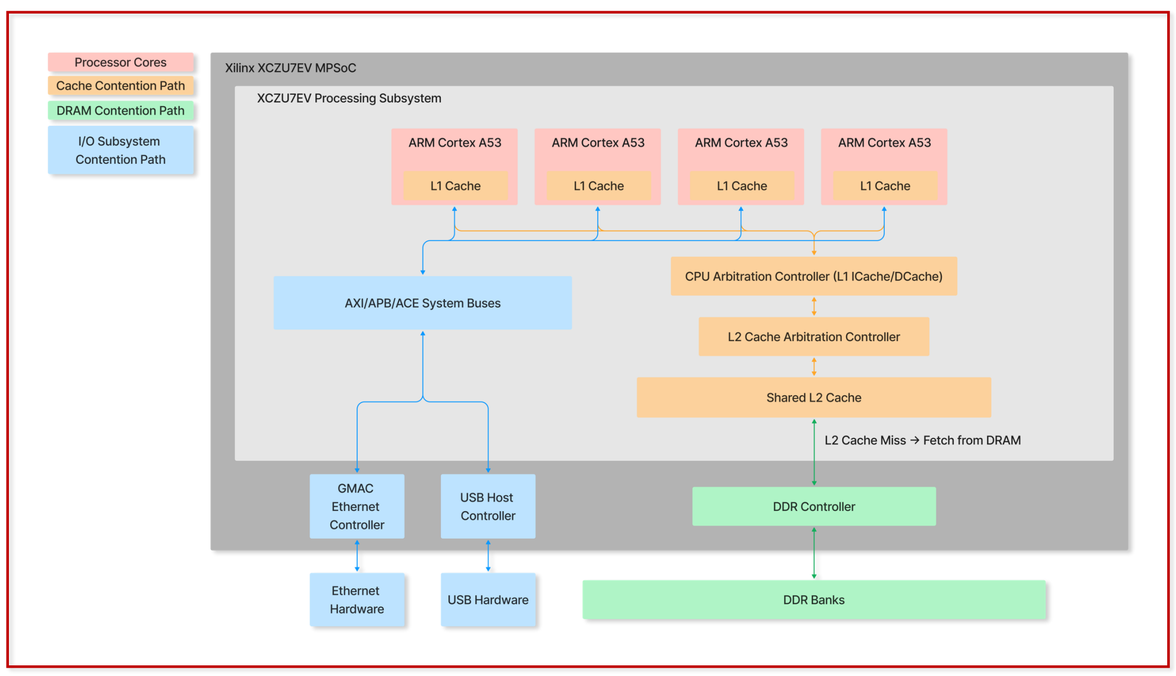

Figure 2: XCZU7EV MPSoC Resource Contention Paths

Test Results

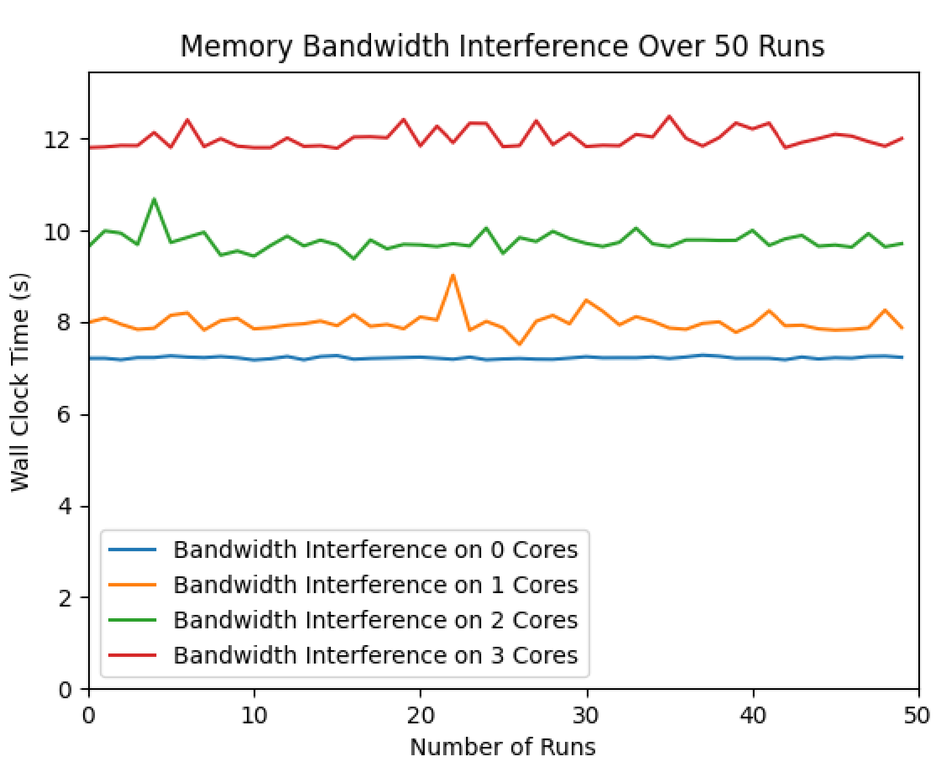

Figure 3: Memory Bandwidth Test Results Graph

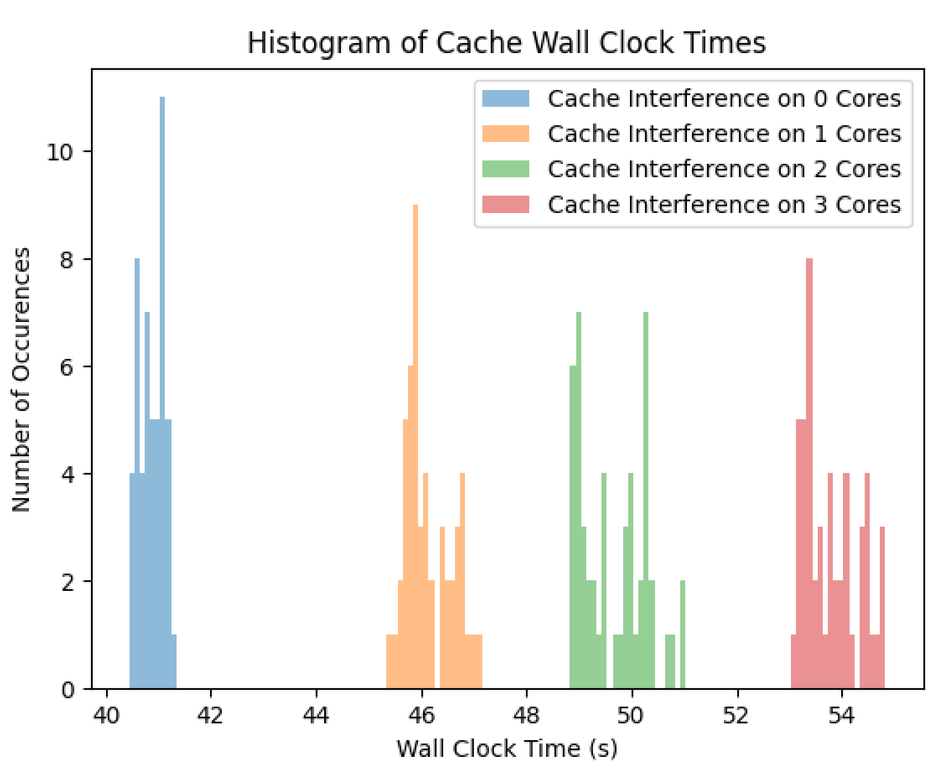

Figure 4: Cache Test Results Graph

These are graphs generated using results from our tool. They show distinct interference from the generators and show that the interference stacks across cores. The results also capture the change in standard deviation, outliers, etc., between runs when more interference is being generated. These results allow for easy analysis of execution time and a visual representation of the architectural interference between supposedly "separate" domains on the hardware.

Industry Standards – *FAA (Federal Aviation Administration) AC 20-193, IEEE (Institute of Electrical and Electronics Engineers) Code of Ethics, CAST-32A, SAE (Society of Automotive Engineers) Aerospace Standards, RCTA/DO-178C, ASTM (American Society for Testing and Standards), POSIX (Portable Operating System Interface), ARINC 653, FACE (Future Airborne Capability Environment).*