A Hardware Event Logger on an FPGA Using a NIOS-V Processor and Linux to Study Task Timing

Teresa Co
Columbia University
New York, USA
tc3499@columbia.edu

Handong He
Columbia University
New York, USA
hh3152@columbia.edu

Xiao Lu
Columbia University
New York, USA
xl3586@columbia.edu

Abstract

Accurate measurement of system behavior and performance is important in embedded systems. Software-based timing methods are often affected by delays and system overhead. This project proposes a hardware-based event logging system implemented on an FPGA to provide precise timing measurements. A clock-cycle counter records timestamps, a NIOS-V processor generates events, and a Linux system analyzes the data. This approach improves accuracy and provides a flexible platform for studying system performance.

1 Introduction and Motivation

Understanding system behavior and performance is essential in embedded systems. Developers often need to measure execution time, scheduling behavior, and system efficiency. Software-based profiling tools can introduce measurement errors due to interrupts and operating system overhead. Hardware-based timing avoids these issues by recording time directly at the clock-cycle level. FPGA-based event logging systems have demonstrated high precision in capturing real-time behavior [1]. Event logging is also widely used in embedded systems for monitoring and debugging [2]. This project builds on these ideas to design a simple and effective hardware event logger.

2 Proposed Work

In this project, we will design and implement a hardware-based event logging system on an FPGA. The system will include a clock-cycle counter and an event buffer to record timestamps of system events. A NIOS-V processor will run a scheduler and generate events such as task start, task end, and context switching. These events will be transmitted to the hardware logger through a memory-mapped interface. In addition, we will develop a Linux-based program that retrieves and analyzes the recorded data. The analysis will focus on measuring execution time, scheduling frequency, and overall system behavior. Finally, we will evaluate the effectiveness of this hardware-based approach by comparing it with traditional software-based timing methods.

3 Project Objectives

The main objective of this project is to design and build a complete system for analyzing performance using hardware-based timing. This includes implementing a hardware module capable of recording precise timestamps, developing firmware that generates meaningful system events, and building a software tool that processes and analyzes the collected data. The project also aims to evaluate system performance and provide insights into scheduling behavior and execution efficiency.

4 System Architecture

The system is composed of three main layers: hardware, processor, and analysis. The hardware layer, implemented on the FPGA, includes a clock counter that provides precise timing, an event buffer that stores timestamped records, and a memory-mapped interface that enables communication with the processor. The processor layer consists of a NIOS-V processor that runs a simple scheduler and generates events such as task start, task completion, and context switching. The analysis layer is implemented on a Linux system, where a software program retrieves the event data and computes performance metrics such as execution time and scheduling frequency [3].

5 Implementation Plan

The implementation of this project will follow several structured phases. The first phase focuses on hardware design, where the clock counter and event buffer will be designed and verified through simulation. The second phase involves deploying the hardware design onto the FPGA platform and testing its functionality. The third phase includes developing firmware for the NIOS-V processor to generate and send events to the hardware logger. The fourth phase focuses on building a Linux-based program to collect and store event data. The final phase involves analyzing the collected data to evaluate system performance and validate the effectiveness of the design.

6 Project Milestones

The project will progress through a series of milestones that define key stages of development. The first milestone involves setting up the development environment and finalizing the system design. The second milestone focuses on implementing and testing the FPGA-based hardware event logger. The third milestone involves developing firmware for the NIOS-V processor to generate system events. The fourth milestone focuses on integrating the Linux-based analysis tool for data collection and processing. The fifth milestone involves evaluating system performance and analyzing the results. The sixth milestone includes refining and optimizing the system through debugging and improvements. The final milestone involves preparing the final report and summarizing the outcomes of the project.

7 Technical Rationale

Hardware-based logging provides precise timing measurements because it operates independently of software execution and avoids delays caused by system overhead [1]. FPGA platforms are widely used for high-performance event processing and data acquisition due to their flexibility and efficiency [3]. By combining hardware-based

measurement with software-based analysis, this system achieves both high accuracy and flexibility in evaluating system performance.

8 Expected Outcomes

This project is expected to produce a fully functional FPGA-based event logging system, along with firmware and software tools that support data collection and analysis. The system will provide accurate measurements of execution time and system behavior, offering insights into scheduling efficiency and performance characteristics.

9 Challenges and Risks

Several challenges may arise during the project, including limited FPGA memory for storing events, potential bottlenecks in data transfer between the FPGA and Linux system, and issues related to clock synchronization. Additionally, the complexity of FPGA and NIOS-V development tools may require additional time for learning and debugging. These challenges will be addressed through careful design, testing, and iterative improvements.

10 References

References

[1] N. Penneman et al., "An FPGA-based real-time event sampler," in *Proc. Int. Symp. Applied Reconfigurable Computing (ARC)*, 2010, pp. 243–254.
[2] A. De La Piedra et al., "Secure event logging in sensor networks," *Computers & Mathematics with Applications*, vol. 65, no. 5, pp. 762–773, 2013.
[3] Y. Bai et al., "Overview and future developments of the FPGA-based DAQ of COMPASS," *Journal of Instrumentation*, vol. 11, no. 02, 2016.