Design Document: A Simple Hardware Event Logger on FPGA for Measuring Task Timing on NIOS-V

Teresa Co (tc3499), Handong He (hh3152), Xiao Lu (xl3586)
Columbia University — CSEE 4840 Embedded System Design — Spring 2026

Contents

1 Introduction

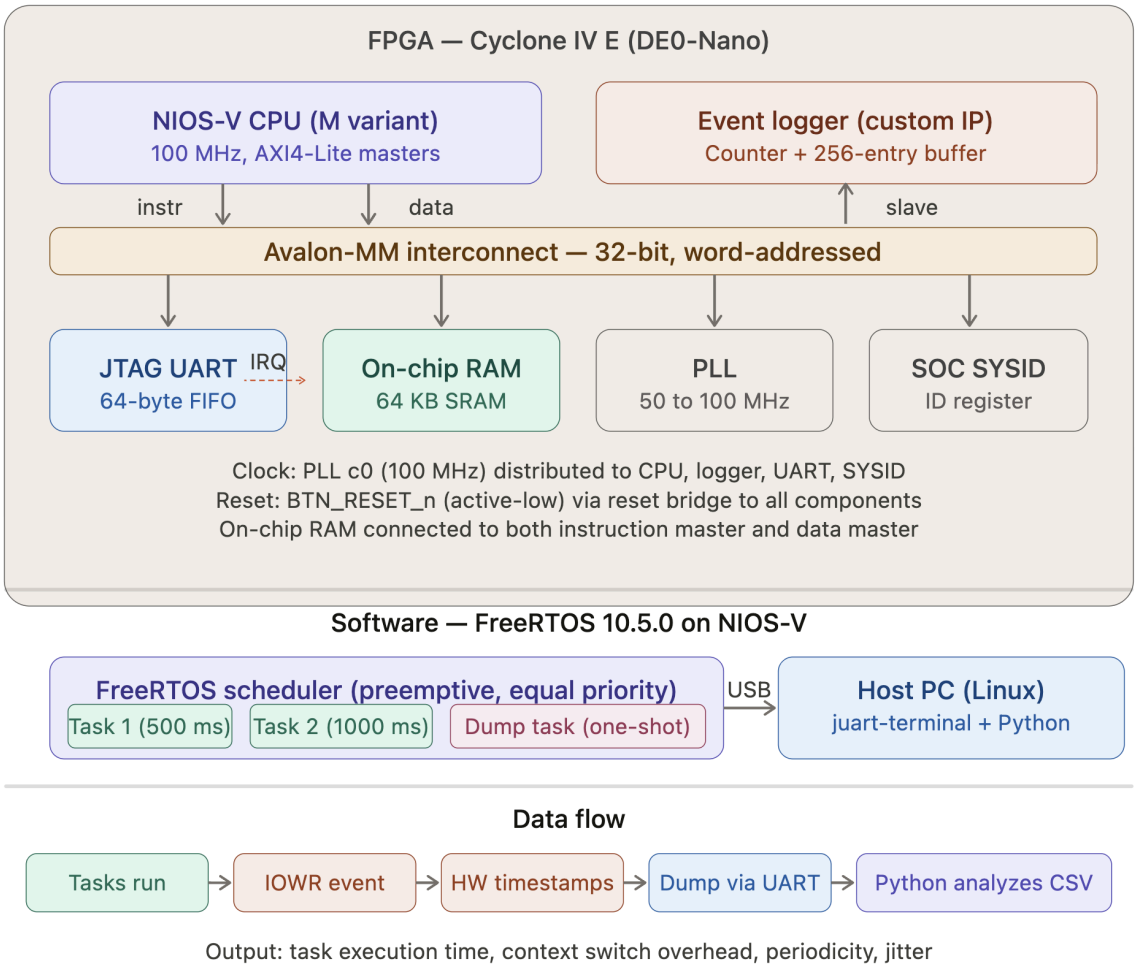
This document describes the design of a hardware-based event logging system for measuring task timing on a NIOS-V soft-core processor running FreeRTOS. The system is implemented on a Terasic DE0-Nano development board using an Intel Cyclone IV E FPGA.

Software-based timing methods (e.g., reading the mtime CSR) introduce measurement overhead of tens to hundreds of CPU cycles per call and are susceptible to interrupt-induced jitter. Our hardware event logger eliminates this problem: the CPU writes a single 32-bit word to a memory-mapped register (one store instruction, ~2 cycles of overhead), and the hardware autonomously captures a cycle-accurate timestamp and stores the event in an on-chip buffer. This provides 10 ns resolution at 100 MHz with minimal intrusion on the system under test.

The system is deliberately kept simple: we removed the external SDRAM controller, EPCS flash, serial UART, buttons, and DIP switches from the original reference design, replacing external memory with 64 KB of on-chip RAM. This minimizes non-deterministic timing effects and allows us to focus on correctness and clarity of measurement.

2 System Block Diagram

The system consists of three layers: a hardware layer on the FPGA that captures and timestamps events, a processor layer where the NIOS-V CPU runs FreeRTOS with instrumented tasks, and an analysis layer on the host PC that processes the logged data.



The NIOS-V CPU fetches instructions and data from the 64 KB on-chip RAM. FreeRTOS schedules two periodic tasks. At each task start and task end, the software writes a single 32-bit word to the event logger's write register. The logger hardware captures the current cycle count, packs it with the event data, and stores the 64-bit entry in its internal ring buffer. After the measurement period, a low-priority dump task reads the buffer contents and outputs them as CSV text through the JTAG UART to the host PC.

3 Algorithms

3.1 Hardware: Event Capture

The event logger implements a simple capture-and-store algorithm entirely in hardware. A 32-bit

free-running counter increments every clock cycle (100 MHz). When the CPU performs a write to register 0, the hardware executes the following in a single clock cycle:

```
on write to EVENT_WRITE register:
    if buffer is not full:
        buffer[wr_ptr] = {cycle_counter, event_type, task_id}
        wr_ptr = wr_ptr + 1
        entry_count = entry_count + 1
    else:
        overflow_flag = 1
```

The buffer operates as a linear buffer with overflow detection (not ring-mode). When full, subsequent writes are silently dropped and the overflow flag is set. This ensures data integrity: no valid entries are ever overwritten.

3.2 Software: FreeRTOS Task Instrumentation

Two periodic tasks are created with equal priority. Each task logs TASK_START at entry and TASK_END before calling vTaskDelay(). A third low-priority dump task sleeps for 10 seconds, then suspends the scheduler and reads out all buffered events as CSV over JTAG UART.

```
void task1(void *p) {
    while (1) {
        IOWR(LOGGER_BASE, 0, (TASK_START << 8) | 1);
        /* measured work */
        IOWR(LOGGER_BASE, 0, (TASK_END << 8) | 1);
        vTaskDelay(pdMS_TO_TICKS(500));
    }
}
```

3.3 Host: Timing Analysis

The Python analysis script parses the CSV dump and computes: (a) per-task execution time = TASK_END timestamp − TASK_START timestamp; (b) context switch overhead = next TASK_START timestamp − previous TASK_END timestamp for consecutive events on different tasks; (c) task periodicity = time between successive TASK_START events for the same task; (d) jitter = max − min of execution times across runs.

4 Resource Budgets

The DE0-Nano's Cyclone IV E (EP4CE22F17C6) provides 22,320 logic elements, 594 KB of embedded memory, and 66 M9K blocks. Our resource consumption:

Component	Logic Elements	M9K Blocks	On-Chip RAM

NIOS-V CPU (M variant)	~4,500	8	—
On-Chip Memory (64 KB)	~20	16	64 KB
Event Logger (256×64-bit)	~150	4	2 KB
JTAG UART	~400	2	—
PLL + Interconnect	~800	0	—
Total (estimated)	~5,870	30	66 KB
Available	22,320	66	594 KB
Utilization	~26%	~45%	~11%

Table 1: Estimated FPGA resource usage.

Event buffer memory budget:

Parameter	**Value**
Entry width	64 bits (32-bit timestamp + 16-bit event + 16-bit padding)
Buffer depth	256 entries

Total buffer size	256 × 8 bytes = 2,048 bytes (2 KB)
Maximum recording time	~42.9 seconds (32-bit counter at 100 MHz)
Events before overflow	256

Table 2: Event buffer memory budget.

5 The Hardware/Software Interface

The event logger is mapped into the NIOS-V address space as an Avalon-MM slave at base address 0x0002_0040. It exposes five 32-bit word-addressed registers:

Offse	**Name**	**Access**	**Description**

t			
0	EVENT_WRITE	Write-only	[15:8] event_type, [7:0] task_id
1	EVENT_READ_LO	Read-only	Timestamp (32-bit cycle count)
2	EVENT_READ_HI	Read-only	[15:8] event_type, [7:0] task_id (reading advances read pointer)
3	STATUS	Read-only	[17] full, [16] overflow, [8:0] entry_count
4	CONTROL	Write-only	[0] write 1 to clear buffer + reset counter

Table 3: Event logger register map (base address 0x00020040).

Event Types:

Code	Name	Meaning
0x01	EVT_TASK_START	A task has begun executing
0x02	EVT_TASK_END	A task has finished its work unit
0x03	EVT_CONTEXT_SWITCH	FreeRTOS switched to a different task (future)

Table 4: Event type encoding.

Buffer entry format (64 bits):

Bits [63:32] Timestamp — 32-bit cycle counter value at time of write Bits [31:16] Reserved (zero-padded)
Bits [15:8] Event type (TASK_START=0x01, TASK_END=0x02)
Bits [7:0] Task ID (1 or 2)

Write protocol: CPU executes IOWR(0x00020040, 0, (event_type << 8) | task_id). This is a single store instruction (~2 CPU cycles). The hardware captures cycle_counter in the same clock cycle as the write.

Read protocol: CPU reads register 1 (timestamp), then reads register 2 (event info); the second read automatically advances the read pointer. Repeat for each entry. Register 3 provides the current entry count and overflow status.

Overflow handling: When the buffer is full (256 entries), further writes are silently discarded and bit

[16] of the STATUS register is set. The software checks this flag before analyzing data and reports a warning if set.

6 Software Design

The software runs on the NIOS-V CPU under FreeRTOS 10.5.0. Three tasks are created:

Task	Priority	Stack	Period	Function
task1	2	512 words	500 ms	Short computation loop, logs start/end
task2	2	512 words	1000 ms	Longer computation loop, logs start/end
dump_task	1 (lowest)	1024 words	One-shot	Waits 10 s, dumps buffer via JTAG UART

Table 5: FreeRTOS task configuration.

The dump task outputs data in CSV format between markers --- EVENT LOG START --- and --- EVENT LOG END ---, followed by the total entry count and overflow status. This format is directly parsed by the host-side Python script.

7 Data Analysis

The host-side Python script (analyze_events.py) reads the captured CSV file and computes the following metrics:

Metric	Formula	Unit
Task execution time	TASK_END.timestamp − TASK_START.timestamp	cycles / µs
Context switch overhead	next TASK_START − prev TASK_END	cycles / µs
Task period	TASK_START[n+1] − TASK_START[n]	cycles / ms
Execution jitter	max(exec_time) − min(exec_time)	cycles / µs

Table 6: Computed timing metrics.

All timestamps are converted from cycle counts to physical time using the known clock frequency: 1 cycle = 10 ns at 100 MHz. The script also validates the data by checking that measured task periods match the expected vTaskDelay values (500 ms and 1000 ms), providing confidence that the logger is working correctly.