Improving System Predictability and Performance via Hardware Accelerated Data Structures

Chetan Kumar N G, Sudhanshu Vyas, Joseph Zambreno and Phillip Jones
{ckng,spvyas,phjones,zambreno}@iastate.edu

Introduction

Scheduling overhead is a major limiting factor for implementing real-time systems that use temporally fine-grain dynamic schedulers. The core of a real time operating system (RTOS) is the scheduler, which ensures tasks are completed on time. A high resolution timer is required to distribute CPU load in accordance to a scheduling algorithm's needs, however as time management is performed at finer granularities, operating system overhead increases.

We present a **scalable hardware scheduler architecture for real time systems** that **reduces processing overhead** and **improves timing predictability** of the scheduler. Our novel hardware priority queue design supports insertions in constant time, and removals in O(log n) time.

Figure. 1

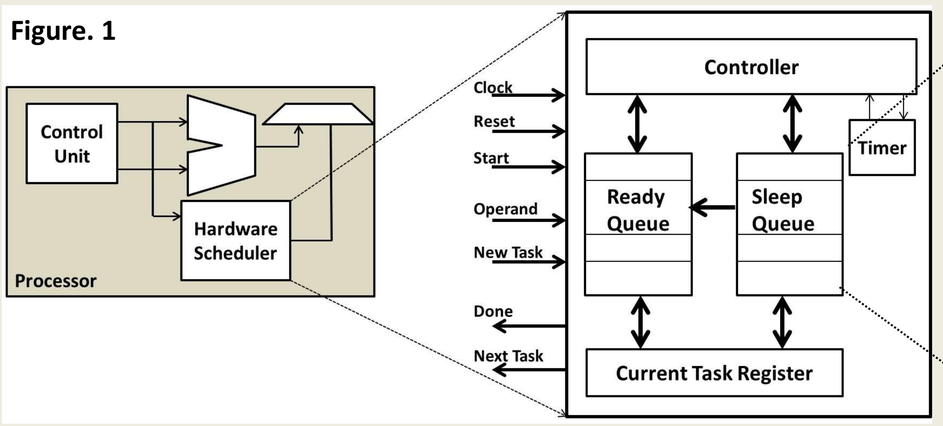
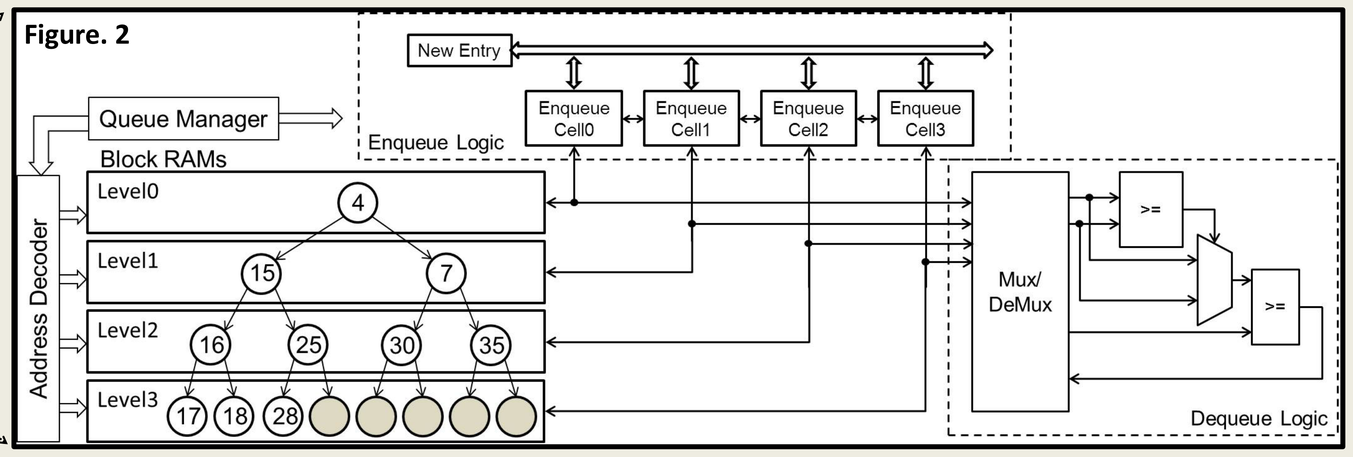


Figure. 2



Hardware Scheduler

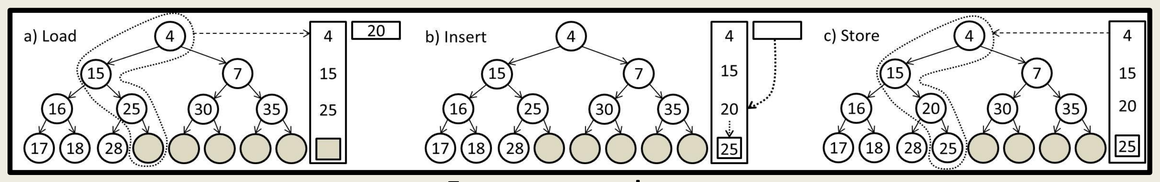
Figure 1 illustrates the high level architecture of our hardware scheduler.

- **Controller:** responsible for the execution of the scheduling algorithm
- **Timer**: keeps accurate high-resolution time.
- **Task queues:** priority queues that keep tasks in sorted order based on their priority (**Ready Queue**) or activation time (**Sleep Queue**).
- **Custom instructions:** extend the processor's instruction set architecture to allow the CPU to interface with the scheduler.

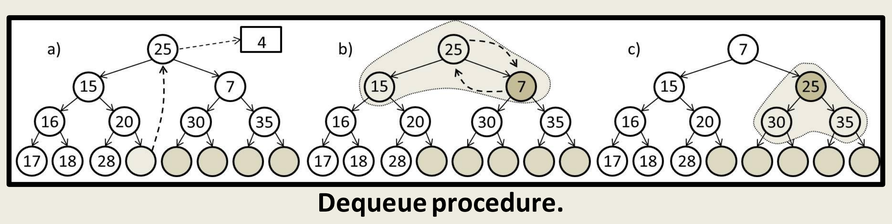
Priority Queue Hardware Architecture

Figure 2 provides further details on the priority queue hardware architecture

- **Data Structure:** a conventional binary heap organization is used to implement a priority queue in hardware.
- **Parallelism:** elements in each level of the heap are stored in separate on-chip memories called Block Rams (BRAMs) to enable parallel access to heap elements.
- **Run time:** enqueue and peek operations take *O(1)* time and dequeue operations take *O(log n)* time.



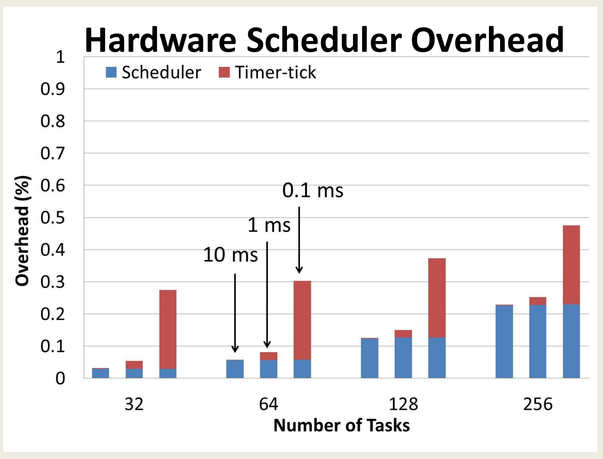
Enqueue procedure.

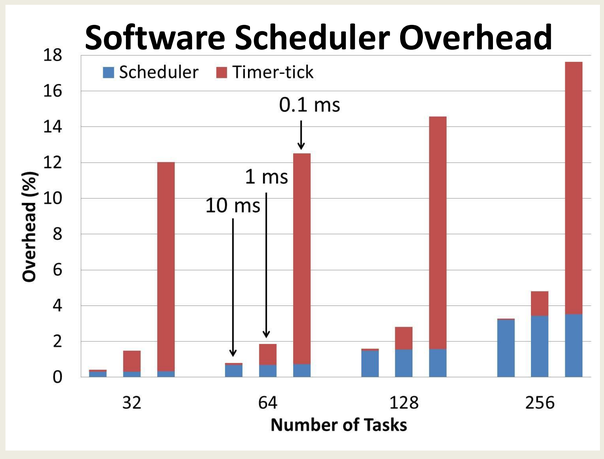


Dequeue procedure.

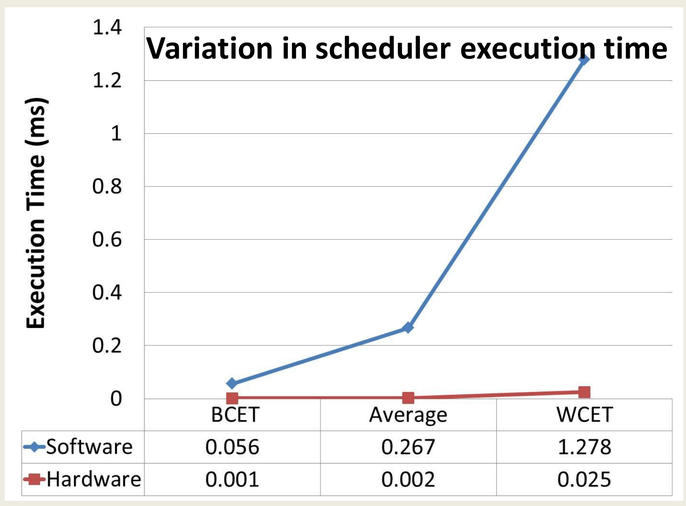
Results and Analysis

- **Scalability**: Supports up to **256 tasks** with high timer tick resolution of **0.1ms**.
- **Overhead**: A 97% reduction in scheduling overhead was obtained after migrating the functionality to hardware.





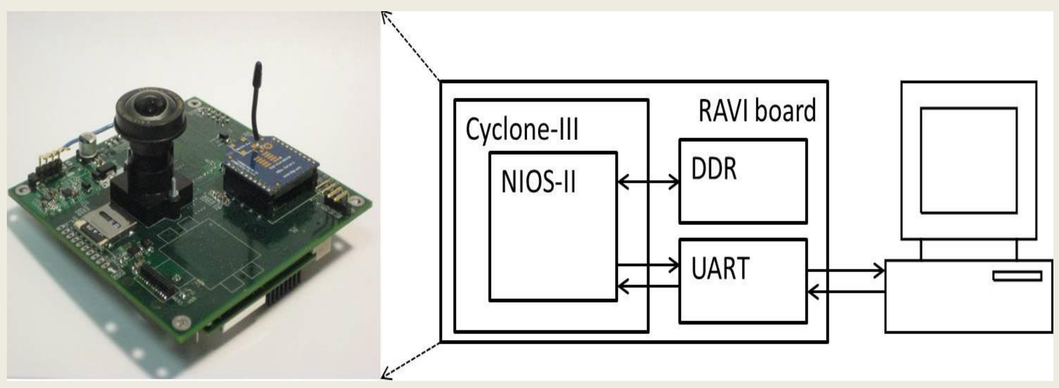
- **Improved Predictability**: Hardware shows 50 times less variability in scheduler execution time, thus providing increased predictability.



	BCET	Average	WCET
Software	0.056	0.267	1.278
Hardware	0.001	0.002	0.025

Platform

Our design was deployed and evaluated on the Reconfigurable Autonomous Vehicle Infrastructure (RAVI) board, an **in-house developed FPGA development platform**. The portions of the RAVI board we used for our experiments included the Cyclone III FPGA, the on-board DDR DRAM and the UART port. The FPGA was used to implement the NIOS-II (Altera's soft-processor), the DDR stored software that was run on the NIOS-II, and the UART port supported data collection. A pictorial description of the setup is shown in figure below.



Future Plans

- Our hardware scheduler is limited to 256 tasks due to on-chip memory constraints. We plan to develop a hybrid SW/HW priority queue architecture that removes this constraint by migrating tasks between on-chip and system memory when the 256 task limit is surpassed.
- Integrate the hardware scheduler into the Linux kernel and evaluate its performance on real world applications.