Integration of Conjoined Cyber-Physical System Properties

Phillip Jones Joseph Zambreno
{phjones,zambreno}@iastate.edu

Ron Cytron Christopher Gill
{cytron,cdgill}@wustl.edu

Introduction

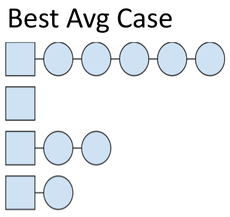
Effective response and adaptation to the physical world, and rigorous management of such behaviors, are mandatory features of cyber-physical systems**.** However, achieving such capabilities across diverse application requirements surpasses the current state of the art in system platforms and tools. Existing systems do not support the expression, integration, and enforcement of such properties that span cyber and physical domains. In this work we are examining mechanisms to enable conjoining of cyber-physical properties within a system through: **1) plastic data structures, 2) tightly coupling SW/HW resources, and 3) integrating system implementation artifacts and control theory.**

1) Plastic Data Structures

While traditional systems often need data structures optimized for fastest average times for **find**, **insert**, and other operations, the more diverse semantics of CPS call for data structures whose optimization criteria can be adapted on the fly at run-time. We have developed *plastic* data structures, which can switch between optimization modes depending on the work being done. The work presented here compares the **Real Time** and **Best Avg Case** modes in terms of the ratio of the average to worst case performance.

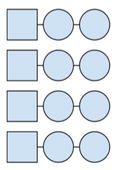
Modes of a Hash Table

Best Avg Case



Best Avg Case: Designed to optimize average performance, this implementation disregards the length of any one chain. When the ratio of nodes to buckets reaches a certain limit, a new table is created and all the nodes are rehashed into the new, larger table. This ratio is known as the load factor.

Real Time

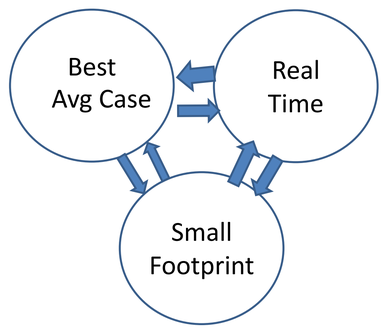


Real Time: This implementation of a hash table has the largest average- to worst-case-time ratio. This implementation places a limit on the maximum number of elements in *any* bucket. In general, real time implementations are slower in terms of average case performance, but better constrain the worst case.

Small Footprint

Small Footprint: The small footprint implementation uses fewer buckets than the other two implementations to conserve memory. A slower implementation in terms of both average case and worst case, this implementation of the hash table is used when memory must be conserved.

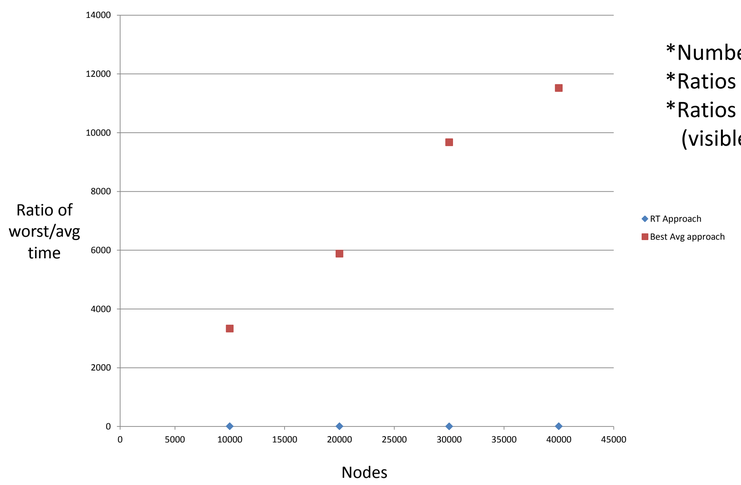
Switching Between Modes



- **Real Time->Best Avg Case**: Consolidate into one table; Requires N puts (N is the #nodes).
- **Real Time->Small Footprint**: Consolidate into a smaller table; Requires B pointer changes to combine all the chains into one or two buckets (B is the # of buckets).
- **Best Avg Case->Small Footprint**: Consolidate into smaller table; Requires B time.
- **Best Avg Case->Real Time**: Clean buckets that need to be cleaned; Worst case N puts.*
- **Small Footprint-> Best Avg Case:** Build new table and rehash all nodes; Requires N puts.
- **Small Footprint-> Real Time:** Clean buckets that need to be cleaned; Worst case N puts. *

Results

The chart below plots the ratio of worst-case-time to average time against the size of a hash table. For real-time (RT), that ratio is ideally 1. Runtime factors beyond our control resulted in ratios closer to 5, but these held steady as the number of nodes increases. However, the ratio is much higher for the best-average method, and becomes worse as the number of nodes increases.



*Numbers are averages taken from multiple trials.
*Ratios for RT method between 4.7—5.2
*Ratios for Best avg method are much higher (visible in plot)

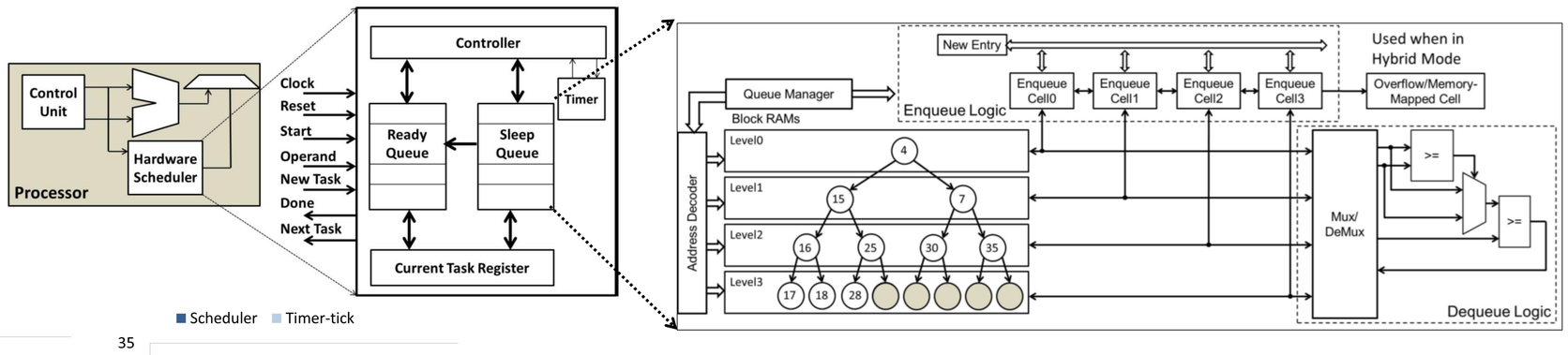
2) Tight Hardware/Software Resource Coupling

In support of plastic data structures, approaches for developing hybrid HW/SW containers are being explored.

- **Performance trade-offs:** provide greater flexible for plastic data structures to meet application requirements.
- **Property guarantees**: make use of hardware mechanisms to help guarantee properties, such as time determinism.

Priority Queue Software/Hardware Hybrid Architecture

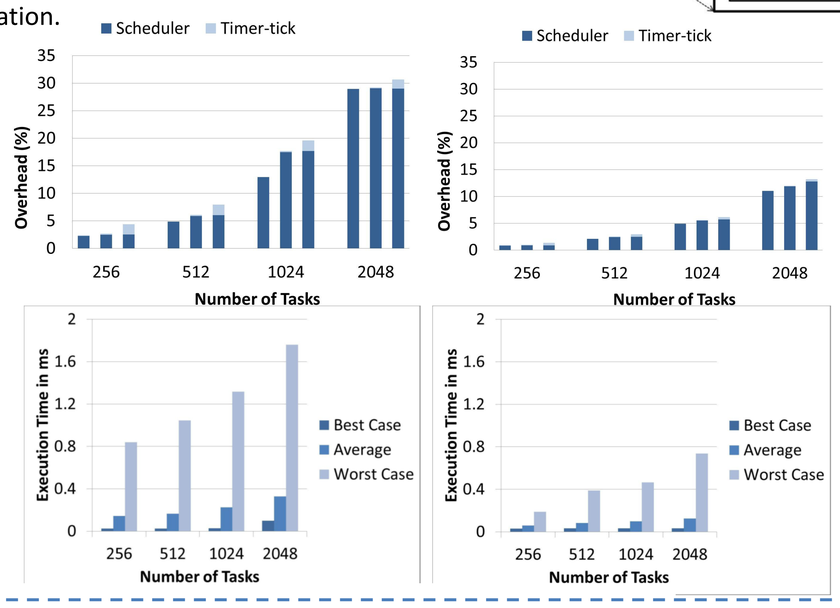
- **Data Structure:** a conventional binary heap organization is used to implement a priority queue in hardware.
- **Parallelism:** each heap level is stored in separate on-chip memories, Block Rams (BRAMs), for parallel access to elements.
- **Run time:** enqueue and peek operations take *O(1)* time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

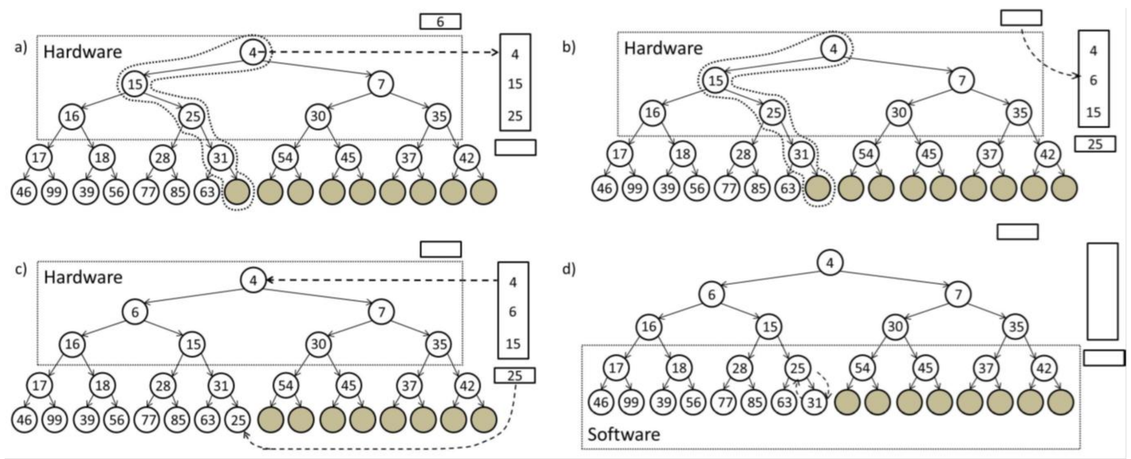


Results and Analysis

Comparing our hybrid priority queue architecture to a software implementation.

- **Scalability**: can support up to **2048 tasks** with high timer tick resolution of **0.1ms**.
- **Overhead**: up to a 50% reduction in scheduling overhead was obtained.
- **Improved Predictability**: up to 2 times less variability in scheduler execution times, thus providing increased predictability.



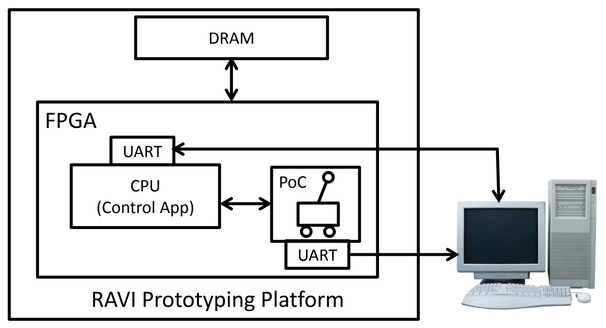


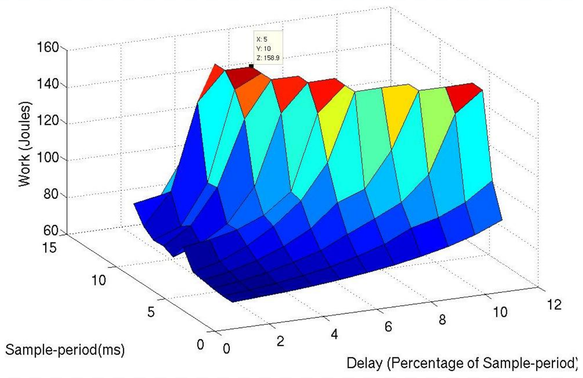
Enqueue procedure: a) ID insertion path, b) parallel element insertion into HW resident portion of heap, c) hand overflow element off to SW, d) complete operation in SW.

3) Integrating Systems Implementation with Control Theory

The design of high performance CPS must take into account implementation artifacts. Our work aims to build upon existing tools such as JitterBug and TrueTime to incorporate artifacts from implementing scheduler and control algorithms (e.g. delay variation) into system stability and performance analysis.

- **Experimentation development**: Matlab-based and Plant-on-Chip (PoC) environments.
- **Characterization**: Currently performing characterization of delay variation on system stability and performance. Within dedicated processor and shared Operating System setups.

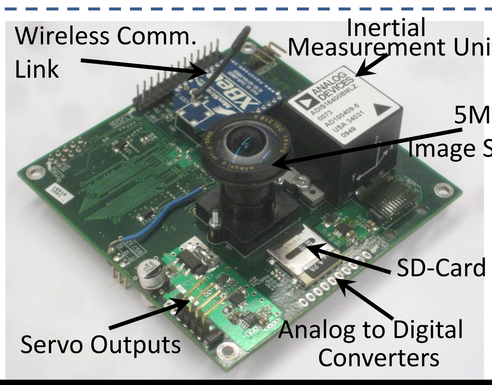




Platform

We have constructed a set of RAVI prototype boards, for experimentation and evaluation of our research. Features on these boards include :

1) an FPGA to support System-on-Chip applications;
2) a high-end inertial measurement unit for integrating physical dynamics of the system;
3) onboard DRAM;
4) a wireless communications module.



Future Plans

- Extend concept of plastic data structures to other container classes (e.g. trees).
- Analyze various “costs” associated with data structure mode transitions.
- Examine low overhead flexible SW/HW hybrid architectures for other data structures.
- Further characterization of interaction between control theory and system implementation artifacts.

Contributing Students

Sudhanshu Vyas, Chetan Kumar N G, Matt Rich (Iowa State); Jonathan Shidal , Josh Levin (Washington University)

Funded by: NSF Grant **1060093**

Washington University in St.Louis
School of Engineering & Applied Science

learn invent impact
Iowa State University
Department of Electrical and Computer Engineering