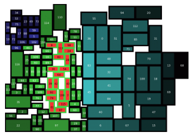
Embedded System Design and Synthesis

Robert Dick

http://ziyang.eecs.northwestern.edu/~dickrp/esds-two-week
Department of Electrical Engineering and Computer Science
Northwestern University

Office at Tsinghua University: 9–310 East Main Building

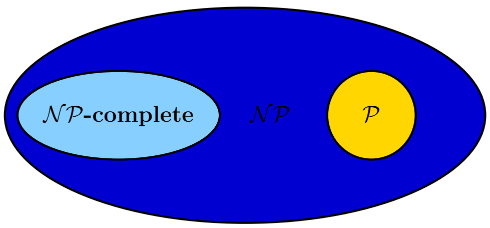




Quiz (page 1)

- Is the monetary size of the general-purpose computing market larger, smaller, or the same as the embedded systems market (one word)?
- What is the time complexity class of linear programming (one word)?
- What is the time complexity class of integer linear programming (one word)?
- What do simulated annealing algorithms do differently at high and low temperatures that permits them to escape local minima?

Basic complexity classes



- $\mathcal{P}$ solvable in polynomial time by a computer (Turing Machine)
- $\mathcal{NP}$ solvable in polynomial time by a nondeterministic computer
- $\mathcal{NP}$-complete converted to other $\mathcal{NP}$-complete problems in polynomial time

Quiz (page 1)

- Is the monetary size of the general-purpose computing market larger, smaller, or the same as the embedded systems market (one word)?
- What is the time complexity class of linear programming (one word)?
- What is the time complexity class of integer linear programming (one word)?
- What do simulated annealing algorithms do differently at high and low temperatures that permits them to escape local minima?

Boltzmann trials

Solution are selected for survival by conducting Boltzmann trials between parents and children.
Given a global temperature T, a solution with cost K beats a solution with cost J with probability:

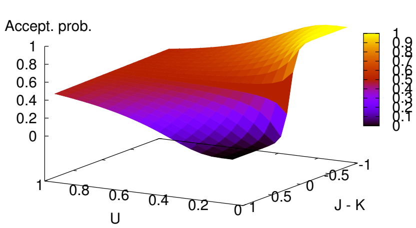
$$\frac{1}{1+e^{(J\text{-}K)/T}}$$

Boltzmann trials

Introduce convenience variable U

$$U(T) = 1 - \frac{1}{T+1}$$
$$U(0) = 0$$
$$T \to 1 \Rightarrow U(T) \to \infty$$

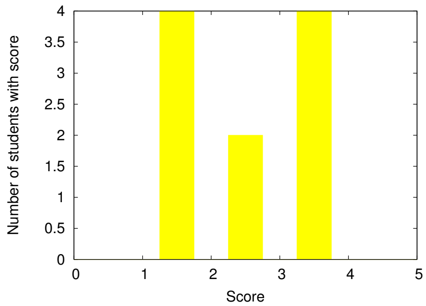
Boltzmann trials



Quiz (page 2)

- You are in the process of designing an embedded system that must prepare a train ticket for a user in a fixed period of time. Ticket request events may occur at any time, but two requests will never be separated by fewer than five seconds. A user should never need to wait more than two seconds from the time they request a ticket to the time the ticket is prepared. The execution time of the ticket preparation task is one second. If you were to map this event-driven system to periodic system, what is the maximum period that can be used while still guaranteeing that the time constraints are met?
- Reliability
  - Name one major lifetime fault process in modern integrated circuits.
  - What things have the most influence over the rate of faults caused by this process?

Quiz One grade distribution



Improving performance

- Some students might get discouraged with the quiz performance
- This is only a small part of the course grade
- Study harder for next quiz
- Keep up on reading and do literature summaries
- Work hard on project
- Do not get discouraged
  - You are as well prepared as many prior students

Essential features of RTOSs

- Provides real-time scheduling algorithms or primatives
- Bounded execution time for OS services
  - Usually implies preemptive kernel
  - E.g., Linux can spend milliseconds handling interrupts, especially disk access

Threads

- Threads vs. processes: Shared vs. unshared resources
- OS impact: Windows vs. Linux
- Hardware impact: MMU

Threads vs. processes

- Threads: Low context switch overhead
- Threads: Sometimes the only real option, depending on hardware
- Processes: Safer, when hardware provides support
- Processes: Can have better performance when IPC limited

Software implementation of schedulers

- TinyOS
- Light-weight threading executive
- μC/OS-II
- Linux
- Static list scheduler

TinyOS

- Most behavior event-driven
- High rate $\rightarrow$ Livelock
- Research schedulers exist

BD threads

- Brian Dean: Microcontroller hacker
- Simple priority-based thread scheduling executive
- Tiny footprint (fine for AVR)
- Low overhead
- No MMU requirements

μC/OS-II

- Similar to BD threads
- More flexible
- Bigger footprint

Old Linux scheduler

- Single run queue
- $\mathcal{O}(n)$ scheduling operation
- Allows dynamic goodness function

$\mathcal{O}(1)$ scheduler in Linux 2.6

- Written by Ingo Molnar
- Splits run queue into two queues prioritized by goodness
- Requires static goodness function
  - No reliance on running process
- Compatible with preemptible kernel

Real-time Linux

- Run Linux as process under real-time executive
- Complicated programming model
- RTAI (Real-Time Application Interface) attempts to simplify
  - Colleagues still have problems at > 18 kHz control period

Real-time operating systems

- Embedded vs. real-time
- Dynamic memory allocation
- Schedulers: General-purpose vs. real-time
- Timers and clocks: Relationship with HW

Introduction

- Real-Time Operating Systems are often used in embedded systems
- They simplify use of hardware, ease management of multiple tasks, and adhere to real-time constraints
- Power is important in many embedded systems with RTOSs
- RTOSs can consume significant amount of power
- They are re-used in many embedded systems
- They impact power consumed by application software
- RTOS power effects influence system-level design

Real-time operating systems (RTOS)

- Interaction between HW and SW
  - Rapid response to interrupts
  - HW interface abstraction
- Interaction between different tasks
  - Communication
  - Synchronization
- Multitasking
  - Ideally fully preemptive
  - Priority-based scheduling
  - Fast context switching
  - Support for real-time clock

General-purpose OS stress

- Good average-case behavior
- Providing many services
- Support for a large number of hardware devices

RTOSs stress

- Predictable service execution times
- Predictable scheduling
- Good worst-case behavior
- Low memory usage
- Speed
- Simplicity

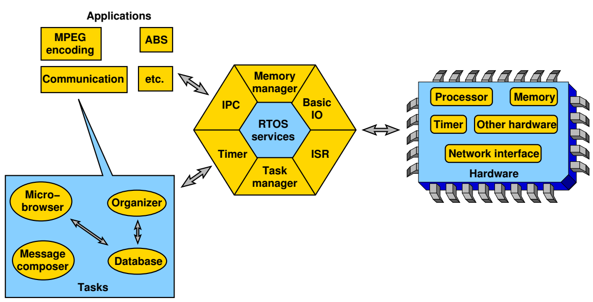
Quiz One discussion | Overview of real-time and embedded operating systems | Embedded application/OS time, power, and energy estimation | Homework
Introduction, motivation, and past work | Examples of energy optimization | Simulation infrastructure | Results

Predictability

- General-purpose computer architecture focuses on average-case
  - Caches
  - Prefetching
  - Speculative execution
- Real-time embedded systems need predictability
  - Disabling or locking caches is common
  - Careful evaluation of worst-case is essential
  - Specialized or static memory management common

Quiz One discussion | Overview of real-time and embedded operating systems | Embedded application/OS time, power, and energy estimation | Homework
Introduction, motivation, and past work | Examples of energy optimization | Simulation infrastructure | Results

RTOS overview



Quiz One discussion | Overview of real-time and embedded operating systems | Embedded application/OS time, power, and energy estimation | Homework
Introduction, motivation, and past work | Examples of energy optimization | Simulation infrastructure | Results

RTOS power consumption

- Used in several low-power embedded systems
- Need for RTOS power analysis
  - Significant power consumption
  - Impacts application software power
  - Re-used across several applications

Quiz One discussion | Overview of real-time and embedded operating systems | Embedded application/OS time, power, and energy estimation | Homework
Introduction, motivation, and past work | Examples of energy optimization | Simulation infrastructure | Results

RTOS and real-time references

- K. Ramamritham and J. Stankovic. Scheduling algorithms and operating systems support for real-time systems. *Proc. IEEE*, 82(1):55–67, January 1994
- Giorgio C. Buttazzo. *Hard Real-Time Computing Systems*. Kluwer Academic Publishers, Boston, 2000

Quiz One discussion | Overview of real-time and embedded operating systems | Embedded application/OS time, power, and energy estimation | Homework
Introduction, motivation, and past work | Examples of energy optimization | Simulation infrastructure | Results

Prior work

- Vivek Tiwari, Sharad Malik, and Andrew Wolfe. Compilation techniques for low energy: An overview. In *Proc. Int. Symp. Low-Power Electronics*, pages 38–39, October 1994
- Y. Li and J. Henkel. A framework for estimating and minimizing energy dissipation of embedded HW/SW systems. In *Proc. Design Automation Conf.*, pages 188–193, June 1998
- J. J. Labrosse. *MicroC/OS-II*. R & D Books, KS, 1998

Quiz One discussion | Overview of real-time and embedded operating systems | Embedded application/OS time, power, and energy estimation | Homework
Introduction, motivation, and past work | Examples of energy optimization | Simulation infrastructure | Results

RTOS power references

Journal version Design Automation Conference 2000 work in the area of RTOS power consumption analysis

- Robert P. Dick, G. Lakshminarayana, A. Raghunathan, and Niraj K. Jha. Analysis of Power Dissipation in Real-Time Operating Systems. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, 22(5):615–627, May 2003

Quiz One discussion | Overview of real-time and embedded operating systems | Embedded application/OS time, power, and energy estimation | Homework
Introduction, motivation, and past work | Examples of energy optimization | Simulation infrastructure | Results

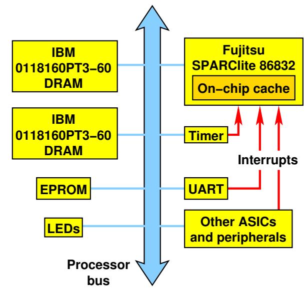
RTOS power references

- K. Baynes, C. Collins, E. Fiterman, B. Ganesh, P. Kohout, C. Smit, T. Zhang, and B. Jacob. The performance and energy consumption of three embedded real-time operating systems. In *Proc. Int. Conf. Compilers, Architecture & Synthesis for Embedded Systems*, pages 203–210, November 2001
- T.-K. Tan, A. Raghunathan, and Niraj K. Jha. EMSIM: An energy simulation framework for an embedded operating system. In *Proc. Int. Symp. Circuits & Systems*, pages 464–467, May 2002

Contributions

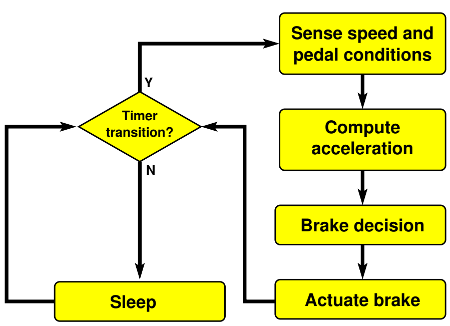
- First detailed power analysis of RTOS
  - Proof of concept later used by others
- Applications
  - Low-power RTOS
  - Energy-efficient software architecture
  - Incorporate RTOS effects in system design

Simulated embedded system

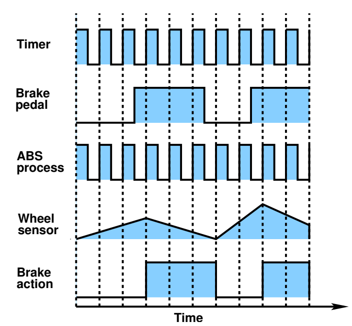


- Easy to add new devices
- Cycle-accurate model
- Fujitsu board support library used in model
- μC/OS-II RTOS used

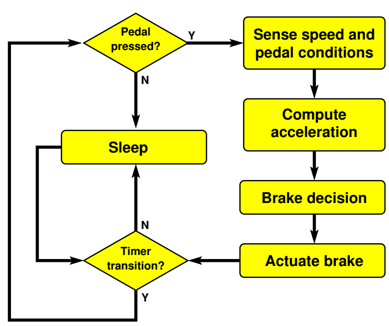
Periodically triggered ABS



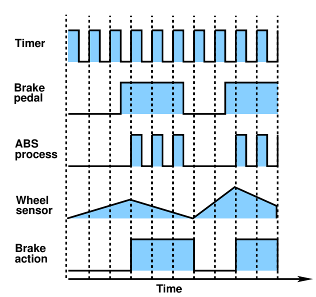
Periodically triggered ABS timing



Selectively triggered ABS

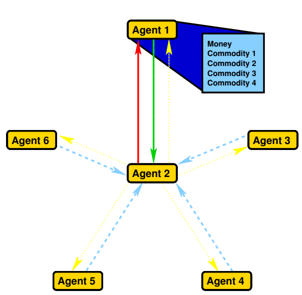


Selectively triggered ABS timing



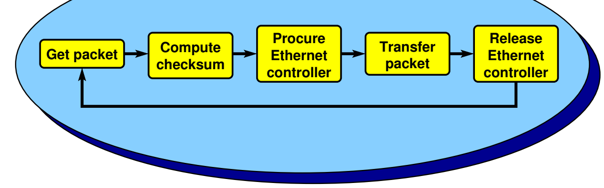
63% reduction in energy and power consumption

Agent example



- Advertise
- Bid
- Offer
- Transfer results

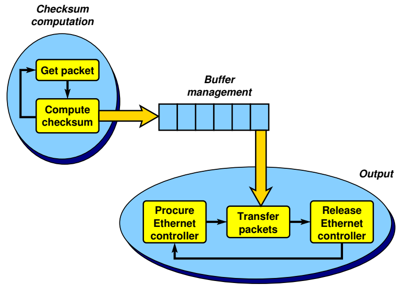
Single task network interface



Checksum computation and output

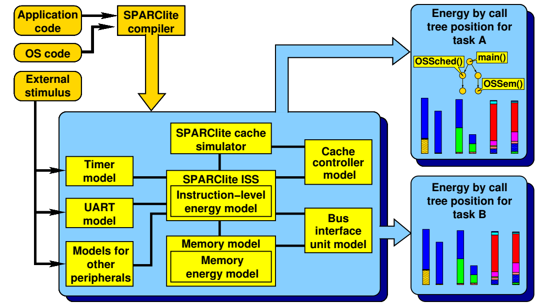
Procuring Ethernet controller has high energy cost

Multi-tasking network interface

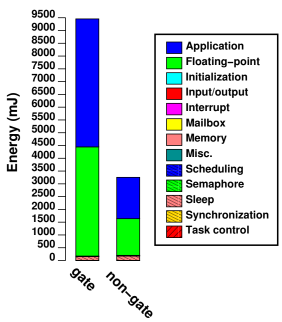


RTOS power analysis suggests process re-organization.
21% reduction in energy consumption. Similar power consumption.

Infrastructure

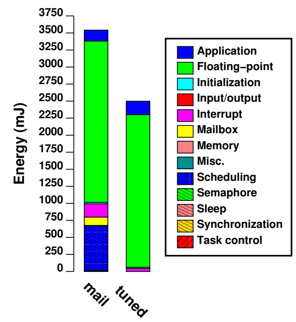


ABS optimization effects



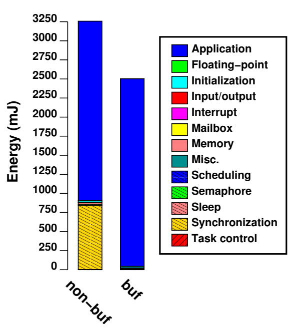
- Redesigned application after using simulator to locate areas where power was wasted
- 63% energy reduction
- 63% power reduction
- RTOS directly accounted for 50% of system energy

Agent optimization effects



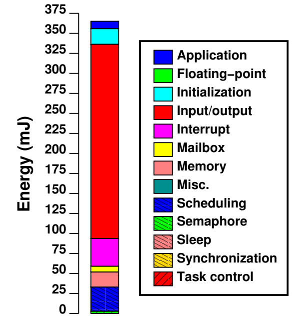
- Mail version used RTOS mailboxes for information transmission
- Tuned version carefully hand-tuned to used shared memory
- Power can be reduced at a cost
  - Increased application software complexity
  - Decreased flexibility

Ethernet optimization effects



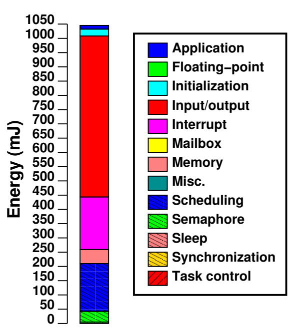
- Determined that synchronization routine cost was high
  - Used RTOS buffering to amortize synchronization costs
- 20.5% energy reduction
- 0.2% power reduction
- RTOS directly accounted for 1% of system energy
  - Energy savings due to improved RTOS use, not reduced RTOS energy

Mailbox example



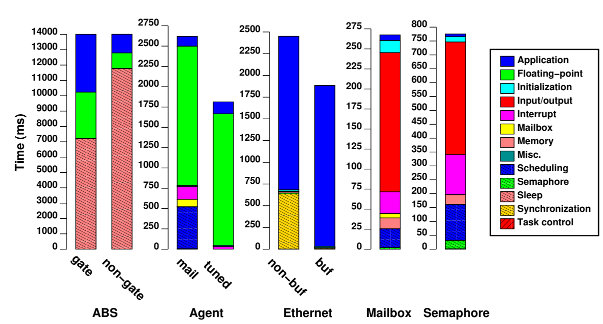
- Rapid mailbox communication between tasks
- RTOS directly accounted for 99% of system energy

Semaphore example



- Semaphores used for task synchronization
- RTOS directly accounted for 98.7% of system energy

Time results



Energy bounds

Service	Minimum energy (μJ)	Maximum energy (μJ)
AgentTask	3.41	4727.88
fptodp	17.46	49.72
BSPInit	3.52	3.52
fstat	16.34	16.34
CPUInit	287.15	287.15
fstat_r	31.26	31.26
GetPsr	0.38	0.55
init_bss	2.86	3.07
GetTbr	0.40	0.53
init_data	4.23	4.37
InitTimer	2.53	2.53
init_timer	18012.10	20347.00
OSCtxSw	46.63	65.65
init_tvecs	1.31	1.31
OSDisableInt	0.84	1.31
...	...	...

Semaphore example hierarchical call tree

		Function	$\frac{\text{Energy}(\mu\text{J})}{\text{invocation}}$	Energy (%)	Time (ms)	Calls
realstart 25.40 mJ total 2.43 %	init_tvecs		1.31	0.00	0.00	1
	init_timer 18.01 mJ total 1.72 %	liteled	4.26	0.00	0.00	1
	startup 7.39 mJ total 0.71 %	do_main	7363.11	0.70	5.57	1
		save_data	5.08	0.00	0.00	1
		init_data	4.23	0.00	0.00	1
		init_bss	2.86	0.00	0.00	1
		cache_on	8.82	0.00	0.01	1
Task1 508.88 mJ total 48.69 %	win_unf_trap		6.09	1.16	9.43	1999
	OSDisableInt		0.98	0.09	0.82	1000
	OSEnableInt		1.07	0.10	0.92	1000
	OSSemPend 104.59 mJ total 10.01 %	win_unf_trap	6.00	0.57	4.56	999
		OSDisableInt	0.94	0.18	1.56	1999
		OSEnableInt	0.94	0.18	1.56	1999
		OSEventTaskWait	13.07	1.25	9.89	999
		OSSched	66.44	6.35	51.95	999
	OSSemPost 9.82 mJ total 0.94 %	OSDisableInt	0.96	0.09	0.78	1000
		OSEnableInt	0.98	0.09	0.81	1000
	OSTimeGet 4.62 mJ total 0.44 %	OSDisableInt	0.84	0.08	0.66	1000
		OSEnableInt	0.98	0.09	0.81	1000
	CPUInit 0.29 mJ total 0.03 %	BSPInit	3.52	0.00	0.00	1
		exceptionHandler	15.51	0.02	0.17	15
	printf 368.07 mJ total 35.22 %	win_unf_trap	6.18	0.59	4.87	1000
		vfprintf	355.04	33.97	257.55	1000

Example power-efficient change to RTOS

- Small changes can greatly improve RTOS power consumption
- μC/OS-II tracks processor loading by incrementing a counter when idle
- However, this is not a good low-power design decision
- NOPs have lower power than add or increment instructions
- Sleep mode has *much* lower power
- Can disable loading counter and use NOPs or sleep mode

Example power-efficient change to RTOS

- Alternatively, can use timer-based sampling
  - Normally NOP or sleep when idle
  - Wake up on timer ticks
  - Sample highest non-timer ISR task
  - If it's the idle task, increment a counter
  - Can dramatically reduce power consumption without losing functionality

RTOS Conclusions

- Demonstrated that RTOS significantly impacts power
- RTOS power analysis can improve application software design
- Applications
  - Low-power RTOS design
  - Energy-efficient software architecture
  - Consider RTOS effects during system design

Reference

Kaushik Ghosh, Bodhisattwa Mukherjee, and Karsten Schwan. A survey of real-time operating systems. Technical report, College of Computing, Georgia Institute of Technology, February 1994

Sensor networking and compression references

- Chee-Yee Chong and Srikanta Kumar. Sensor networks: Evolution, opportunity, and challenges. *Proc. IEEE*, 91(8), August 2003
- Robert P. Dick, Li Shang, and Niraj K. Jha. Power-aware architectural synthesis. In Wai-Kai Chen, editor, *The VLSI Handbook*. CRC Press, 2006

Assignment: Write a short paragraph describing the most important points in both of these articles.