Introduction to Real-Time Systems

ECE 397-1

Northwestern University

Department of Computer Science
Department of Electrical and Computer Engineering

Teachers:	Robert Dick	Peter Dinda
Office:	L477 Tech	338, 1890 Maple Ave.
Email:	dickrp@ece.northwestern.edu	pdinda@cs.northwestern.edu
Phone:	467–2298	467-7859
Webpage:	http://www.ece.northwestern.edu/EXTERNAL/realtime	

Homework index

Goals for lecture

- Lab four?
- Lab six
- Simulation of real-time operating systems
- Impact of modern architectural features

Lab four

- Please email or hand in the write-up for lab assignment four
- Problems? See me.
  - Will need everything from lab four working for lab six

Lab six

- Develop priority-based cooperative scheduler for TinyOS that keeps track of the percentage of idle time.
- Develop a tree routing algorithm for the sensor network.
- Send noise, light, and temperature data to a PPC, via the network root.
- Have motes respond to *send audio samples* and *buzz* commands.
- Play back or display this data on PPCs to verify the that the system functions.

Outline

- Introduction
- Role of real-time OS in embedded system
- Related work and contributions
- Examples of energy optimization
- Simulation infrastructure
- Results
- Conclusions

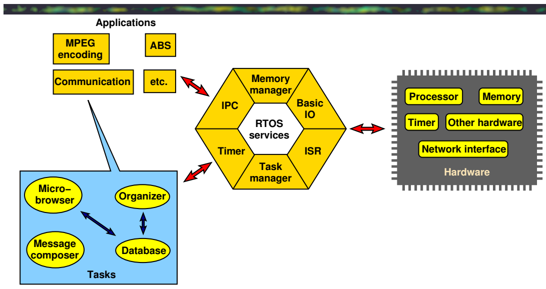
Introduction

- 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.

Introduction

- Real Time Operating Systems important part of embedded systems
  - Abstraction of HW
  - Resource management
  - Meet real-time constraints
- Used in several low-power embedded systems
- Need for RTOS power analysis
  - Significant power consumption
  - Impacts application software power
  - Re-used across several applications

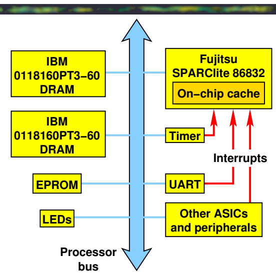
Role of RTOS in embedded system



Related work and contributions

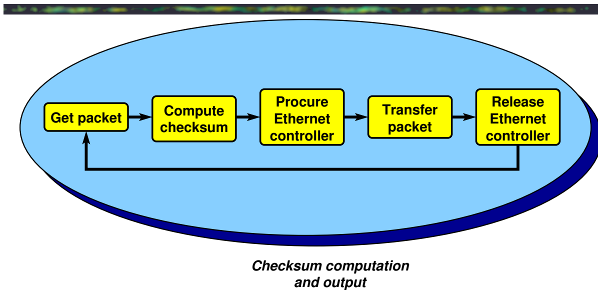
- **Instruction level power analysis**
  V. Tiwari, S. Malik, A. Wolfe, and T.C. Lee, Int. Conf. VLSI Design, 1996
- **System-level power simulation**
  Y. Li and J. Henkel, Design Automation Conf., 1998
- **MicroC/OS-II**: J.J. Labrosse, R & D Books, Lawrence, KS, 1998
- **Our work**
  - First step towards detailed power analysis of RTOS
  - Applications: low-power RTOS, energy-efficient software architecture, incorporate RTOS effects in system design

Simulated embedded system



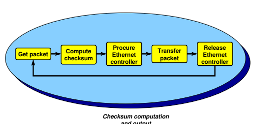
- Easy to add new devices
- Cycle-accurate model
- Fujitsu board support library used in model
- μC/OS-II RTOS used

Single task network interface

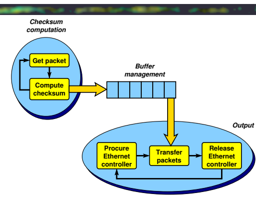


Procuring Ethernet controller has high energy cost

TCP example

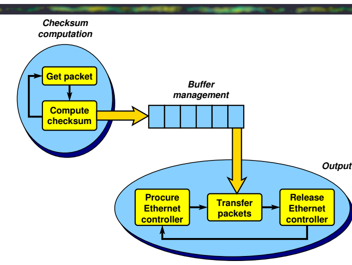


Straight-forward implementation



Multi-task implementation

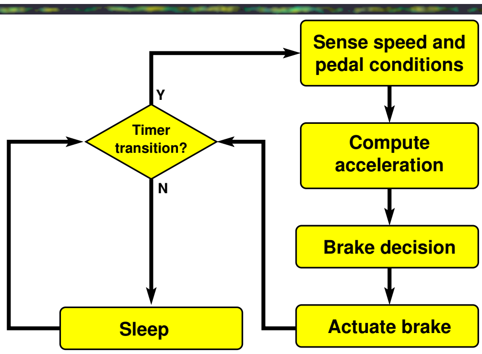
Multi-tasking network interface



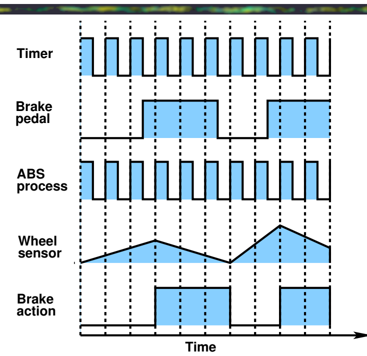
RTOS power analysis used for process re-organization to reduce energy

21% reduction in energy consumption. Similar power consumption.

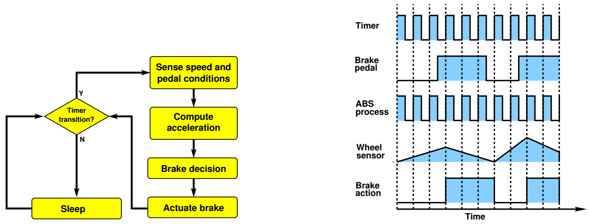
ABS example



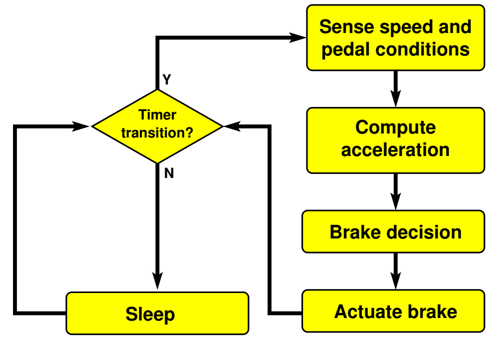
ABS example timing



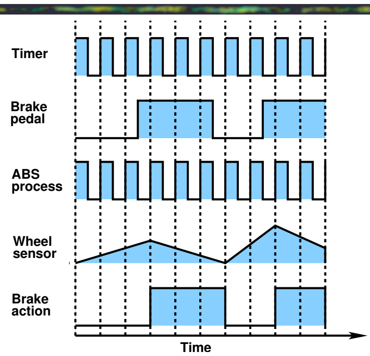
Straight-forward ABS implementation



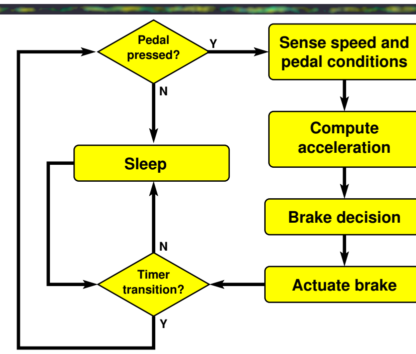
Periodically triggered ABS



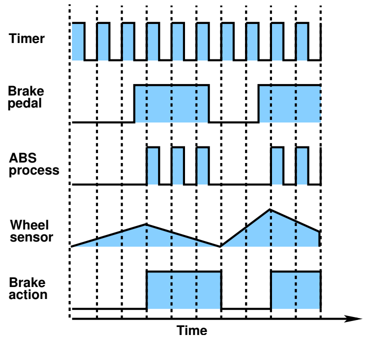
Periodically triggered ABS timing



Selectively triggered ABS

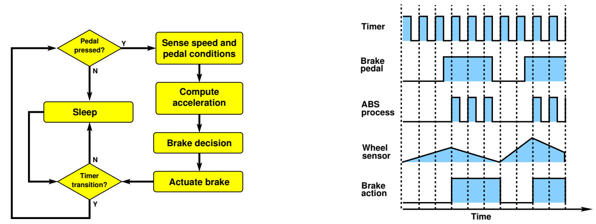


Selectively triggered ABS timing

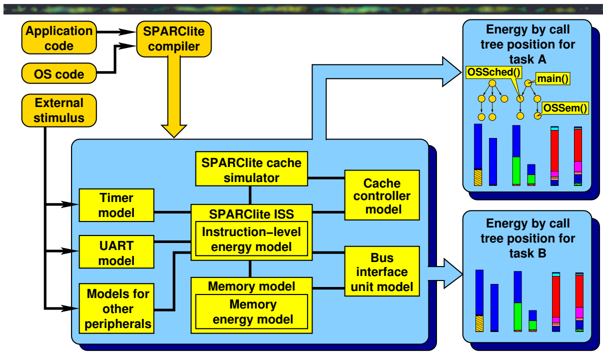


63% reduction in energy and power consumption

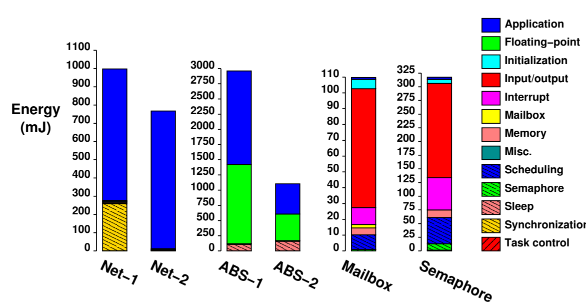
Power-optimized ABS example



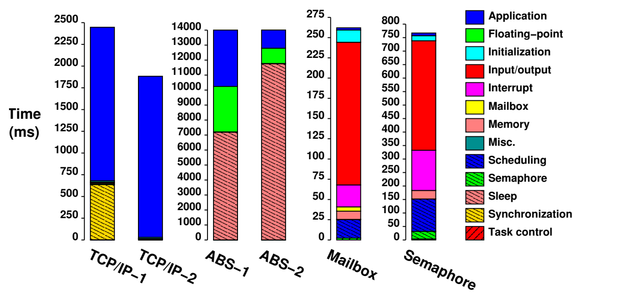
Infrastructure



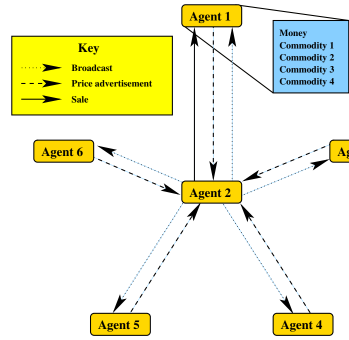
Experimental results



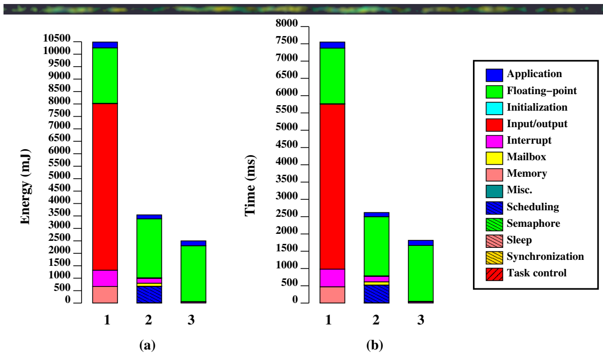
Experimental results – time



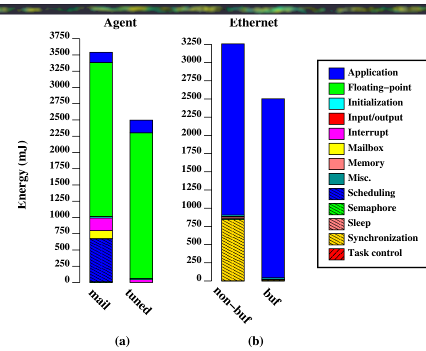
Agent example



Experimental results



Experimental results



Optimization effects

TCP example:

- 20.5% energy reduction
- 0.2% power reduction
- RTOS directly accounted for 1% of system energy

ABS example:

- 63% energy reduction
- 63% power reduction
- RTOS directly accounted for 50% of system energy

Mailbox example: RTOS directly accounted for 99% of system energy

Semaphore example: RTOS directly accounted for 98.7% of system energy

Partial semaphore hierarchical results

		Function	Energy/invocation (uJ)	Energy (%)	Time (mS)	Calls
realstart 6.41 mJ total 2.02 %	init_tvecs		0.41	0.00	0.00	1
	init_timer 5.51 mJ total 1.74 %	liteled	1.31	0.00	0.00	1
	startup 0.90 mJ total 0.28 %	do_main	887.44	0.28	2.18	1
		save_data	1.56	0.00	0.00	1
		init_data	1.31	0.00	0.00	1
		init_bss	0.88	0.00	0.00	1
		cache_on	2.72	0.00	0.01	1
Task1 155.18 mJ total 48.88 %	win_unf_trap		1.90	1.20	9.73	1999
	_OSDisableInt		0.29	0.09	0.78	1000
	_OSEnableInt		0.32	0.10	0.89	1000
	sparcsim_terminate		0.75	0.00	0.00	1
	OSSemPend 31.18 mJ total 9.82 %	win_unf_trap	2.48	0.78	6.33	999
		_OSDisableInt	0.29	0.18	1.59	1999
		_OSEnableInt	0.29	0.18	1.59	1999
		OSEventTaskWait	3.76	1.18	9.22	999
		OSSched	19.07	6.00	47.97	999
	OSSemPost 2.90 mJ total 0.91 %	_OSDisableInt	0.29	0.09	0.78	1000
		_OSEnableInt	0.29	0.09	0.78	1000
	OSTimeGet 1.43 mJ total 0.45 %	_OSDisableInt	0.27	0.08	0.70	1000
		_OSEnableInt	0.29	0.09	0.78	1000
	CPUInit 0.09 mJ total 0.03 %	BSPInit	1.09	0.00	0.00	1
		exceptionHandler	4.77	0.02	0.17	15
	printf 112.90 mJ total 35.56 %	win_unf_trap	2.05	0.65	5.06	1000
		vfprintf	108.89	34.30	258.53	1000

Energy per invocation for μC/OS-II services

Service	Minimum energy (μJ)	Maximum energy (μJ)
OSEventTaskRdy	18.02	20.03
OSEventTaskWait	7.98	9.05
OSEventWaitListInit	20.43	21.16
OSInit	1727.70	1823.26
OSMboxCreate	27.51	28.82
OSMboxPend	7.07	82.91
OSMboxPost	5.82	84.55
OSMemCreate	19.40	19.75
OSMemGet	6.64	8.22
OSMemInit	27.41	27.47
OSMemPut	6.38	7.91
OSQInit	20.10	20.93
OSSched	6.96	52.34
OSSemCreate	27.87	29.04
OSSemPend	6.54	73.64
etc.	etc.	etc.

Conclusions

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

Impact of modern architectural features

- Memory hierarchy
- Bus protocols ISA vs. PCI
- Pipelining
- Superscalar execution
- SIMD
- VLIW

Summary

- Labs
- Simulation of real-time operating systems
- Impact of modern architectural features