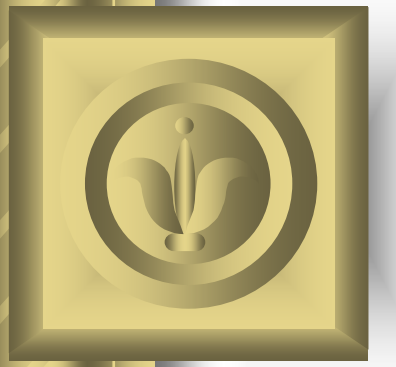


Hardware Support for Priority Inheritance

Bilge E. S. Akgul[+], Vincent J. Mooney[+], Henrik Thane[*] and Pramote Kuacharoen[+]

[+]Center for Research on Embedded Systems and Technology (CREST)
[+]School of Electrical and Computer Engineering
[+]Georgia Institute of Technology, USA

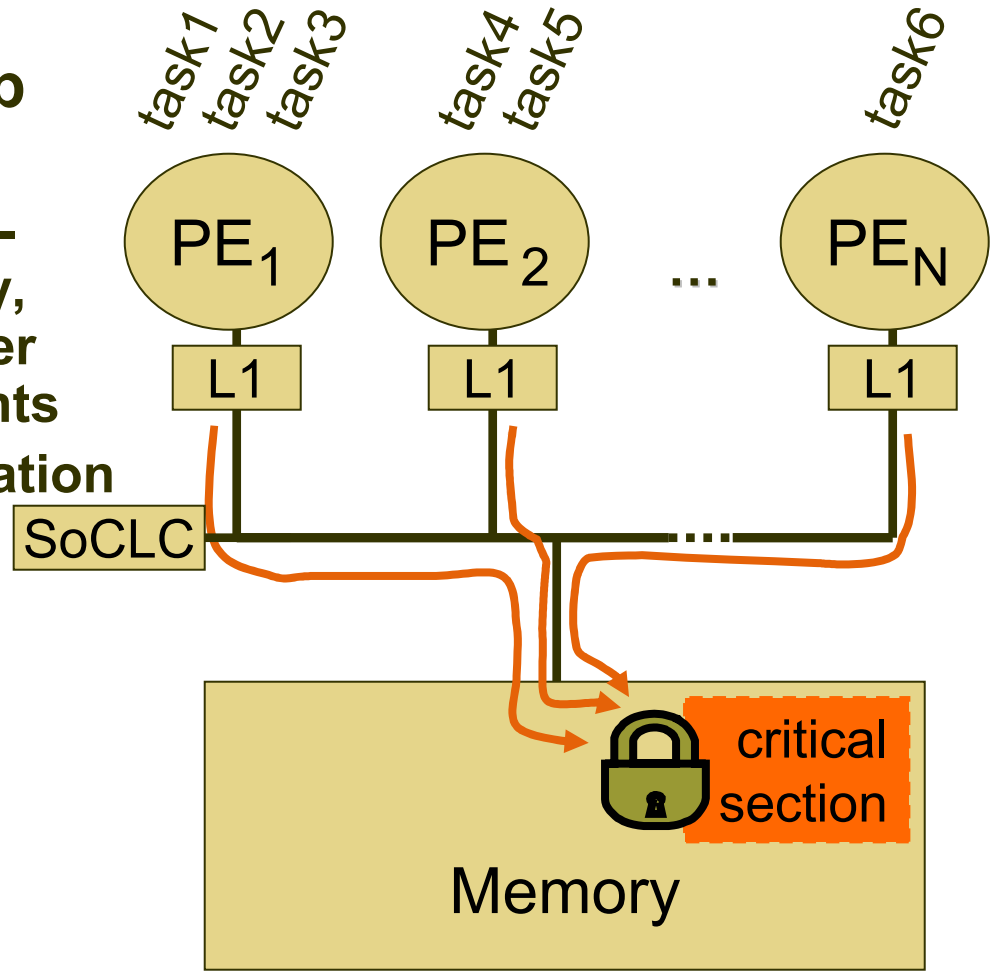
***Malardalen Real-Time Research Center (MRTC)**
***Malardalen University, Sweden**

Outline

- Introduction
  - SoCLC
- Background & Motivation
  - Priority Inheritance
- Related work
- SoCLC with IPCP
- Experiment & Results
- Conclusion

Introduction

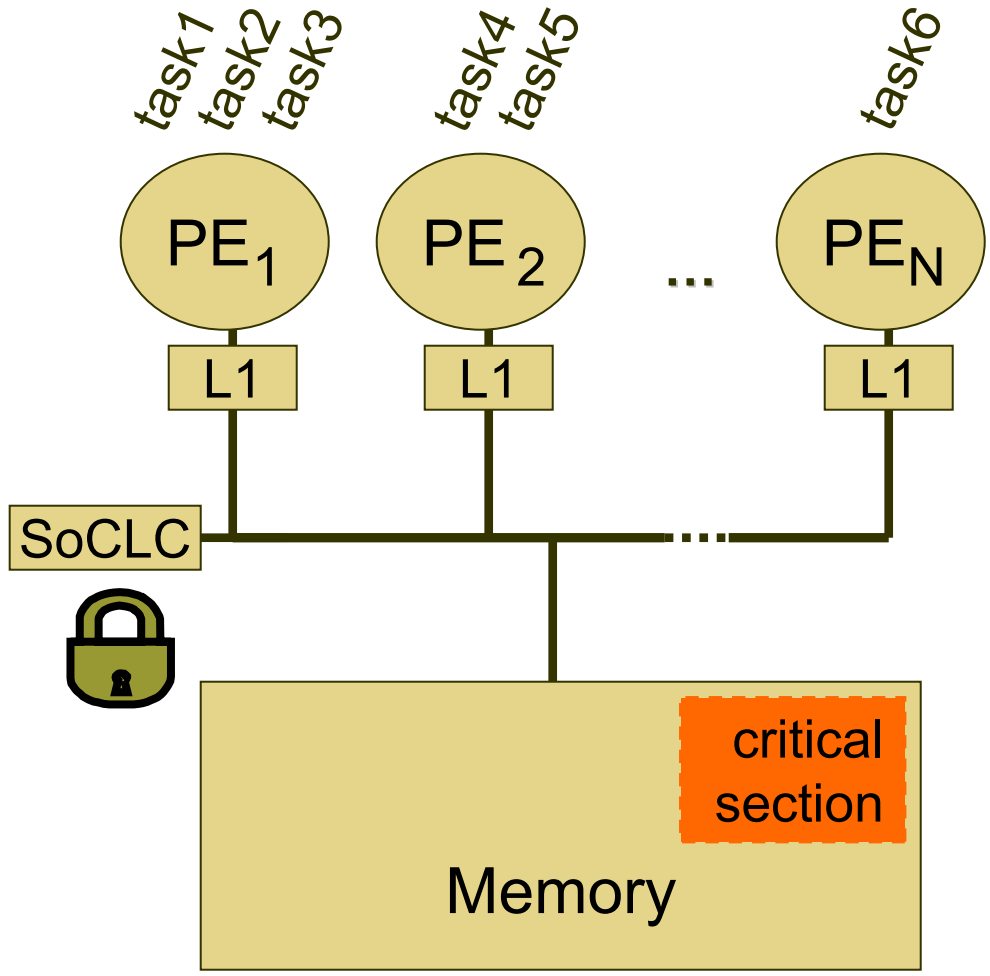
- A system-on-a-chip (SoC) may include
  - Multiprocessors, on-chip shared memory, peripherals and other hardware components
  - Multi-tasking application with a real-time operating system (RTOS)
- Many shared-data structures cause contention



PE: processing element

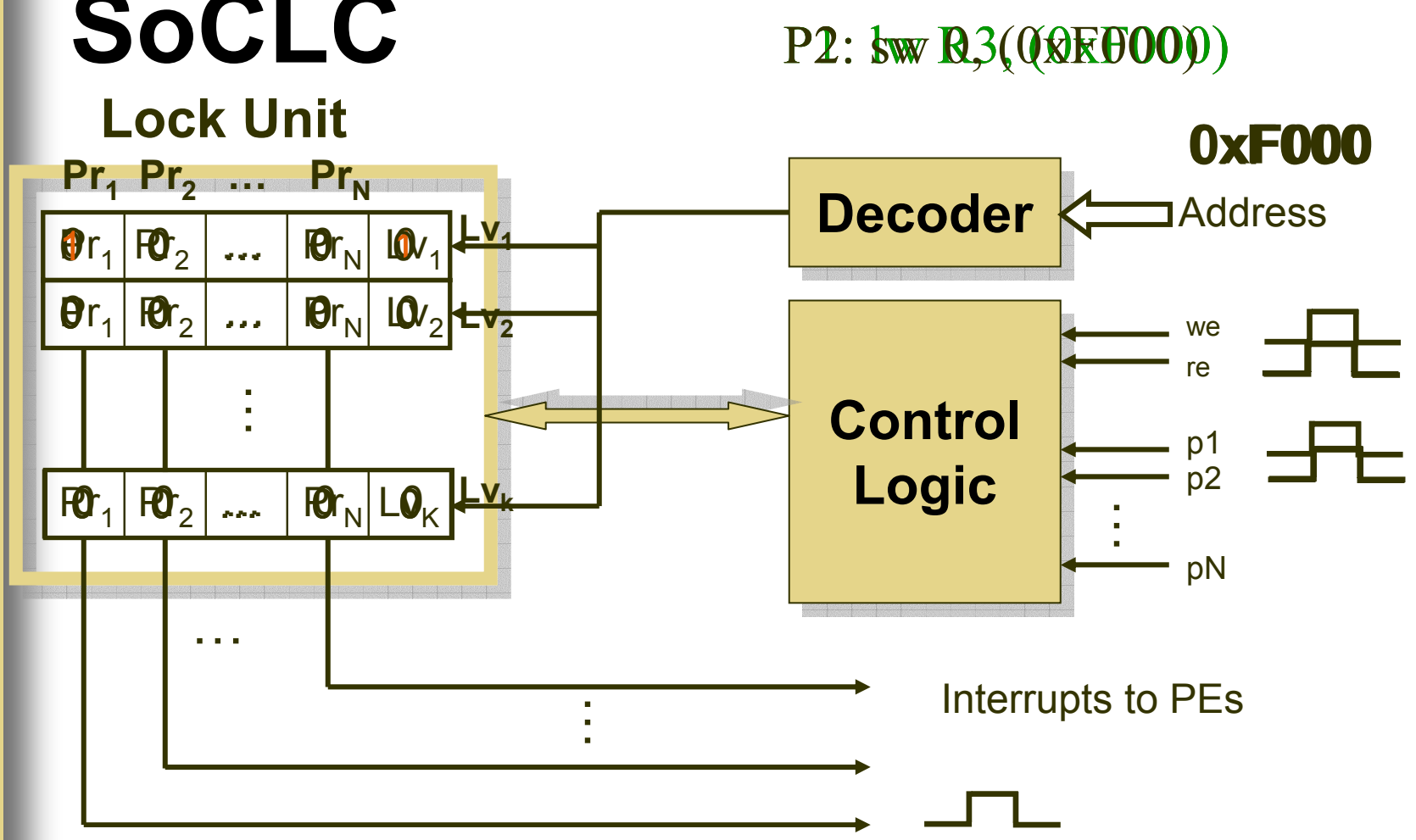
Introduction

- **Solution: move lock variables to a specialized hardware logic**
- **SoC Lock Cache (SoCLC)**



PE: processing element

SoCLC



- 27% to 55% speedups
  - DATE'01, CASES'01, DAES'02

Motivation

- Real-time operating system (RTOS) with a priority-based scheduler
  - Assign a higher priority to a time-critical task with hard real-time requirement
- Problem: If tasks with different priorities share resources ➔ priority inversion may occur
  - May miss real-time deadlines

Motivation

- Priority inheritance in RTOS
  - May affect real-time performance of application tasks
- Objective: To implement hardware support for priority inheritance (via SoCLC) to help RTOS be more predictive and efficient

Priority Inheritance Protocols

- Sha, Rajkumar and Lehoczky (`88)
- Prevents unbounded blocking
  - Running task inherits the highest dynamic priority of all the tasks it blocks
- List of blocked tasks must be saved in a priority queue for each CS
- Maximum blocking time (due to a lower priority task):
  - On each lock, at most once
  - Length of one CS (executed in a lower priority task)
- Still problem: deadlock, chained blockings

Priority Ceiling Protocols

- Sha, Rajkumar and Lehoczky (`90)
- Baker (`91)
- Klein and Ralya (`90)
- Prevent deadlocks and chained blockings
  - Implies that once a process locks its first CS, it can never be blocked by lower priority tasks
- Original priority ceiling protocol (OPCP)
- Immediate priority ceiling protocol (IPCP)
- Each task has a static (default) priority

IPCP vs. OPCP

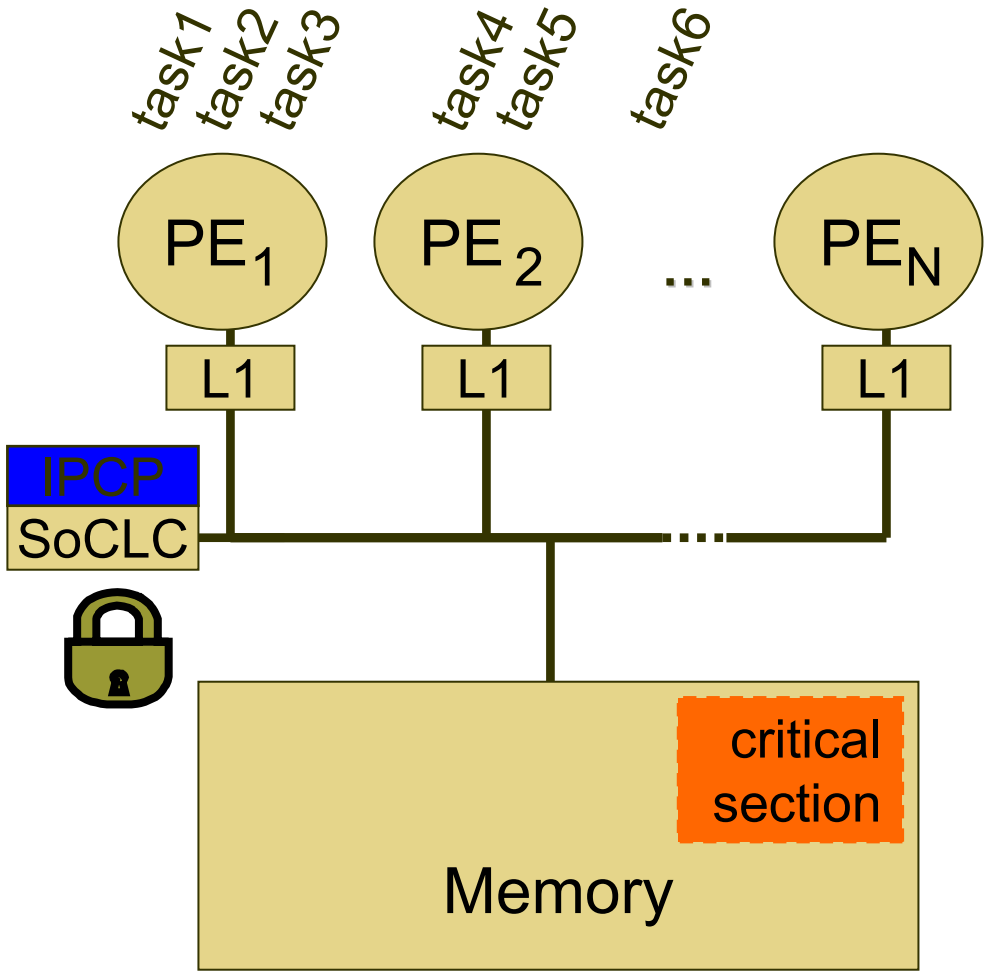
	IPCP	OPCP
Algorithm	simpler	more complex
Ceiling priority	highest priority of any task using CS	highest priority of any task using CS
Dynamic priority	when CS accessed	when higher priority task blocked
# context switches	less	more
Max blocking duration	one CS duration	one CS duration

IPCP is currently used in POSIX, RT Java, Ada

Related Work

- Operating system coprocessors
- Implement various real-time functions in hardware
  - Real Time Unit (RTU), `96
    - Many RTOS functions in hardware
  - Ada TAsking Coprocessor (ATAC), `95
    - It has its own instruction set
    - Implements real-time part of Ada (also Ada rendezvous with basic priority inheritance) in hardware

Our New Approach: SoCLC Priority Inheritance

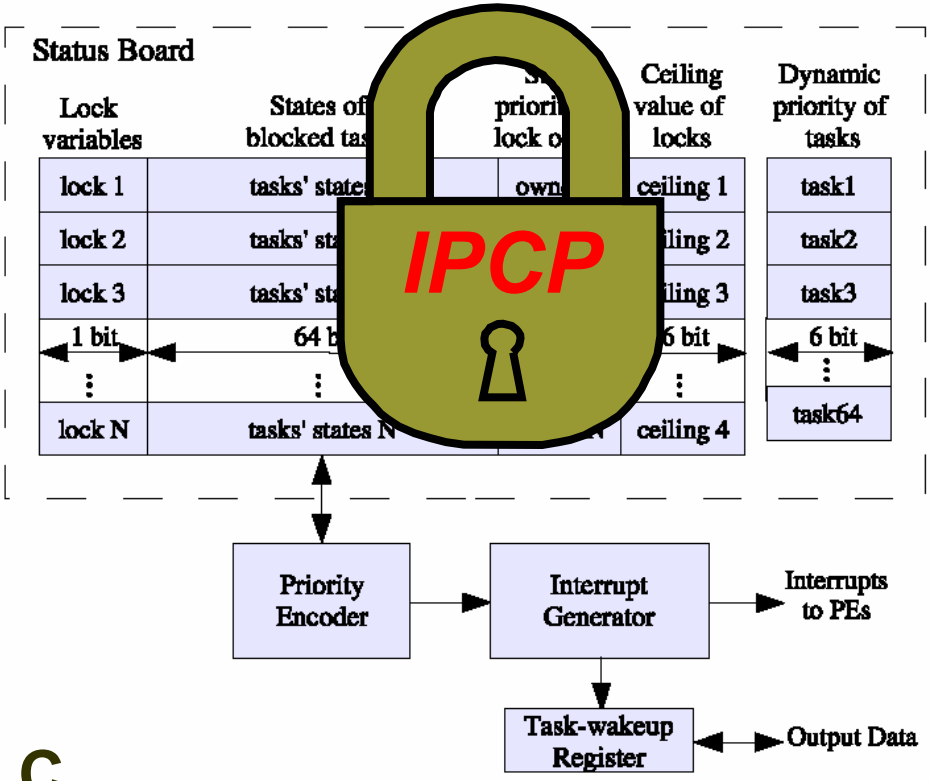


PE: processing element

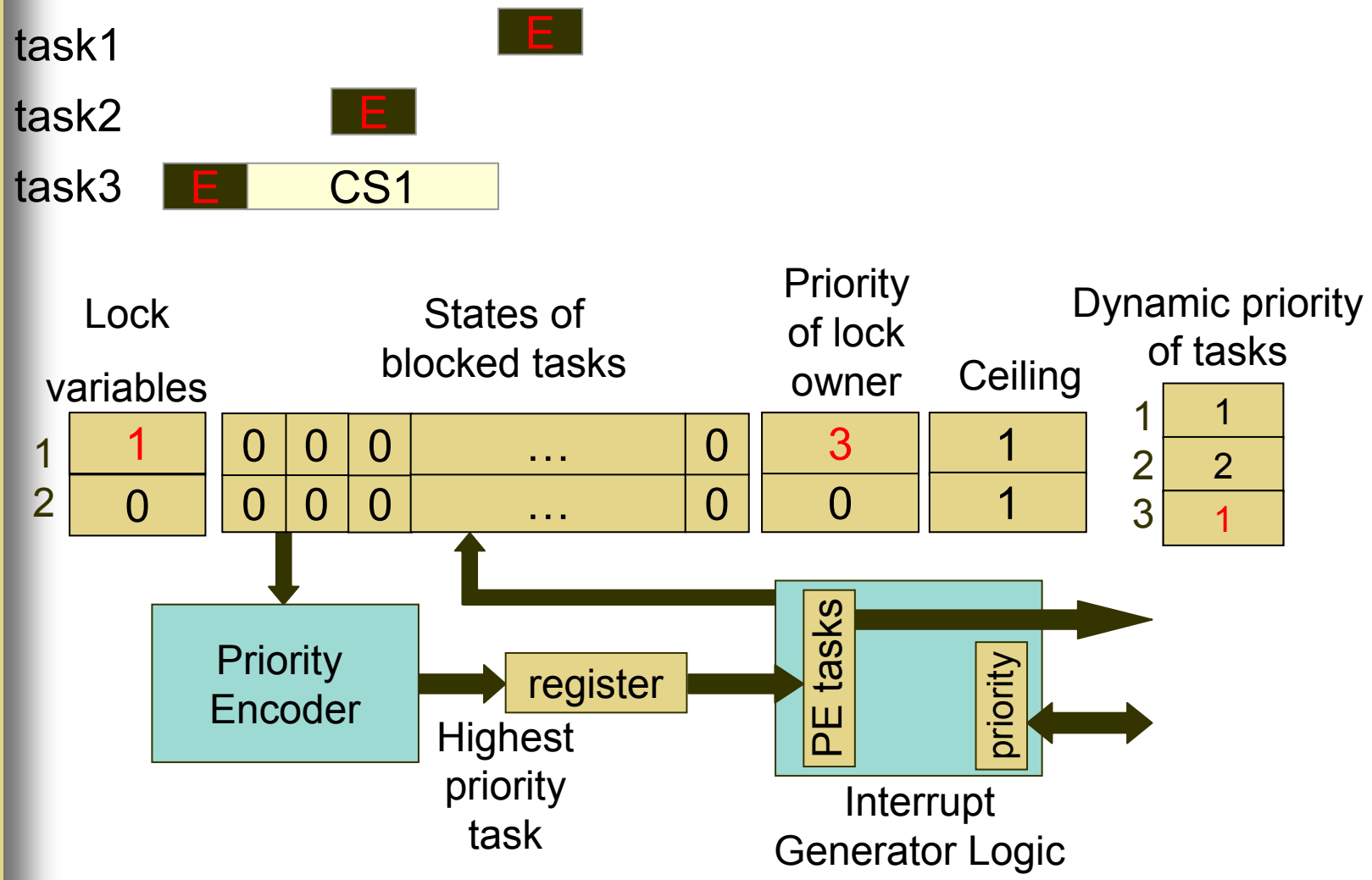
Our New Approach: SoCLC Priority Inheritance

- SoCLC with IPCP
- Ceiling values for every CS used in each task is specified
  - SoCLC needs the ceiling values of locks
- Task priorities are updated by SoCLC in hardware
- Blocked tasks are monitored by SoCLC

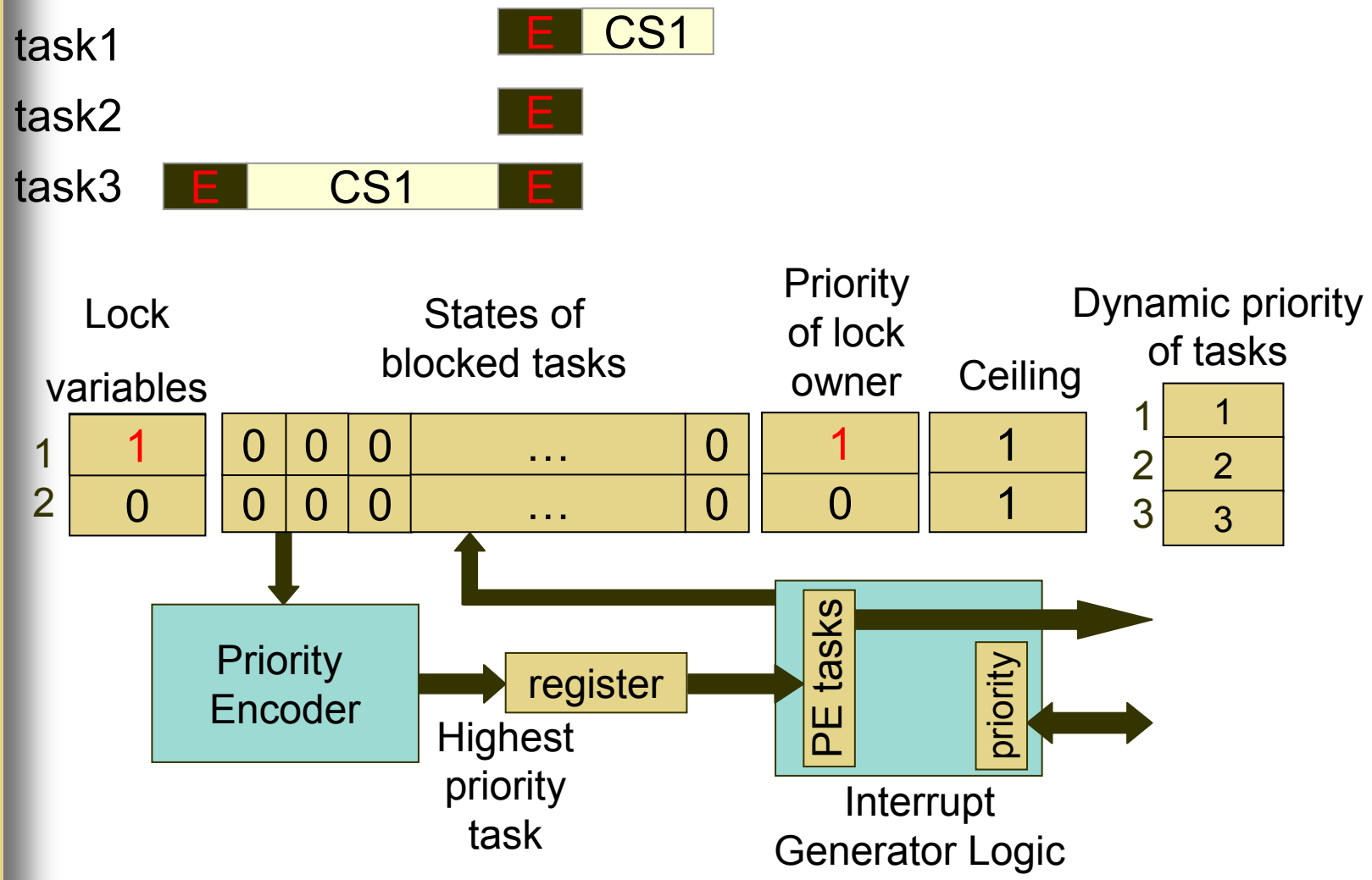
Priority Inheritance Hardware Architecture for a 64-task RTOS



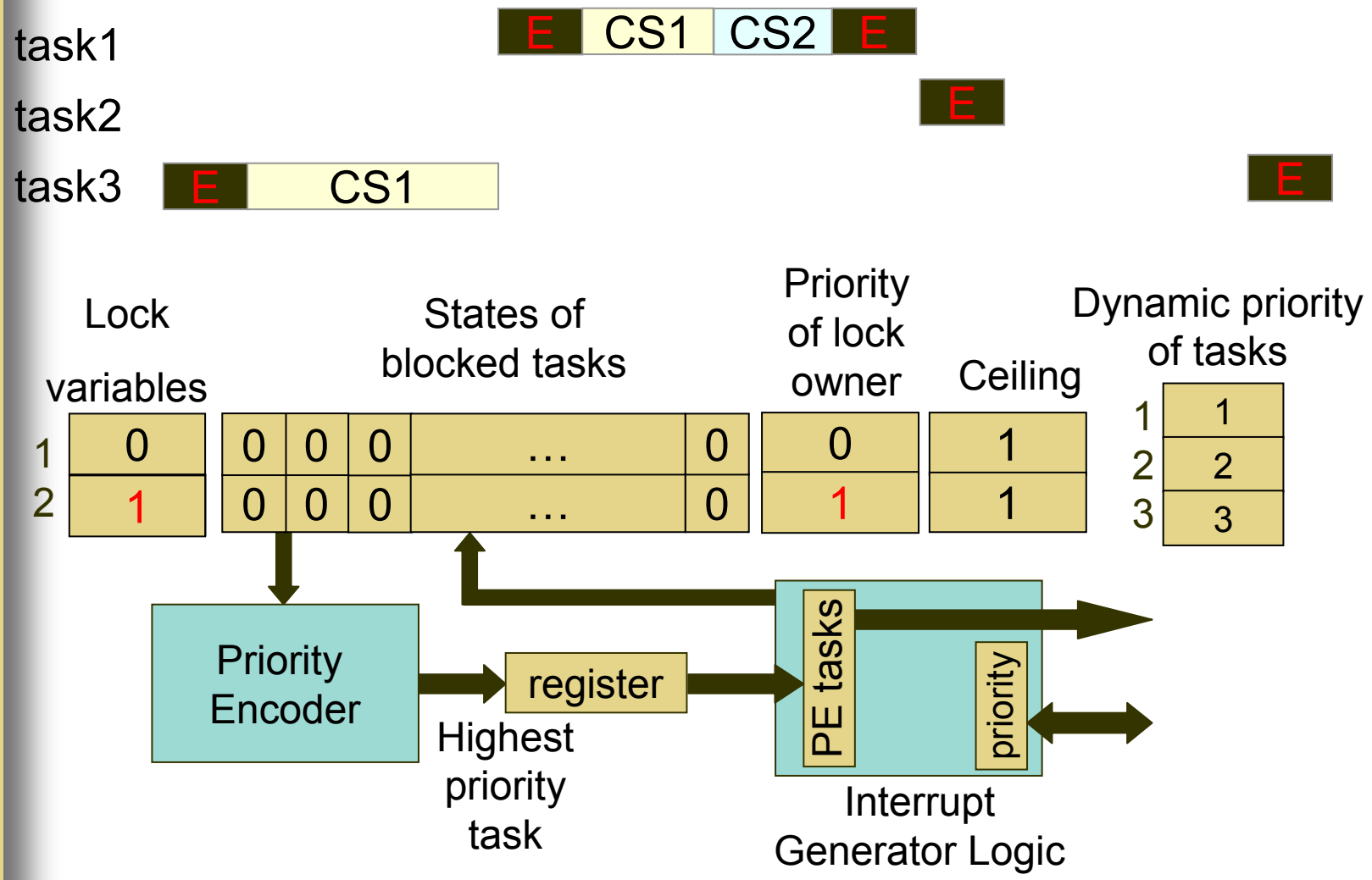
Example


task1
E
task2
E
task3
E
CS1
Lock variables
States of blocked tasks
Priority of lock owner
Ceiling
Dynamic priority of tasks
1
2
1
0
0 0 0 … 0
0 0 0 … 0
3
0
1
1
1
2
2
3
1
Priority Encoder
register
Highest priority task
PE tasks
priority
Interrupt Generator Logic

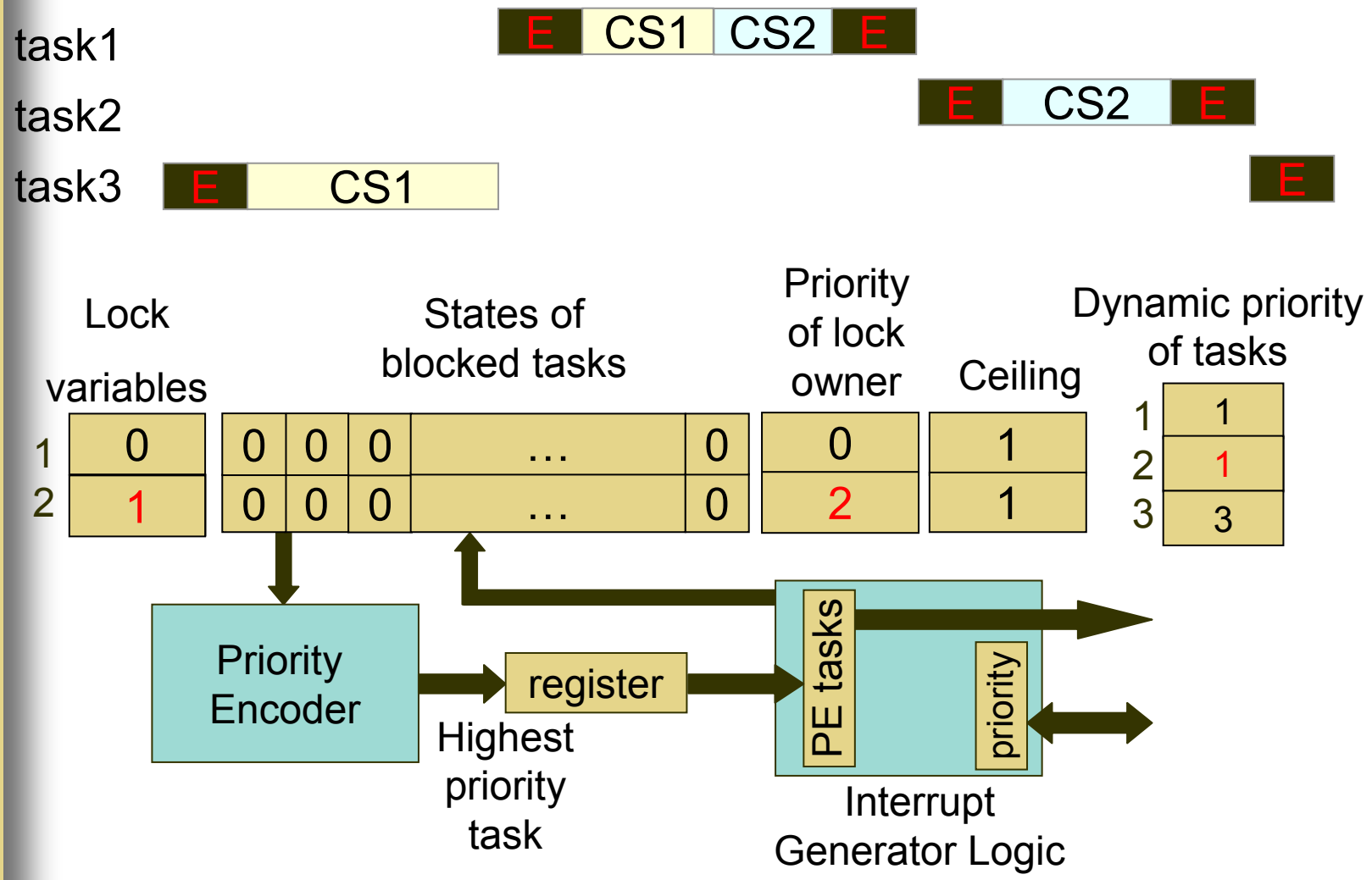
Example


task1
E
CS1
task2
E
task3
E
CS1
E
Lock variables
States of blocked tasks
Priority of lock owner
Ceiling
Dynamic priority of tasks
1
2
1
0
0 0 0 … 0
0 0 0 … 0
1
0
1
1
1
2
2
3
3
Priority Encoder
register
Highest priority task
PE tasks
priority
Interrupt Generator Logic

Example

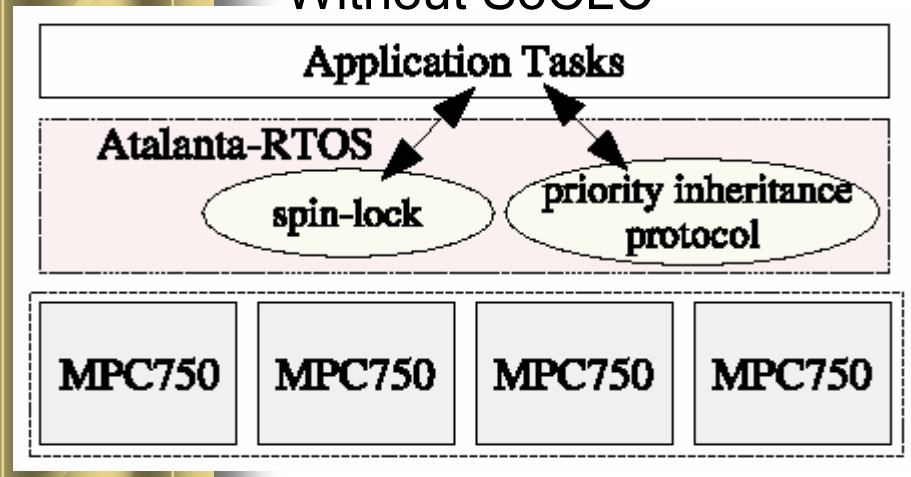

task1
E
CS1
CS2
E
task2
E
task3
E
CS1
E
Lock variables
States of blocked tasks
Priority of lock owner
Ceiling
Dynamic priority of tasks
1
2
0
1
0 0 0 … 0
0 0 0 … 0
0
1
1
1
1
2
3
1
2
3
Priority Encoder
register
Highest priority task
PE tasks
priority
Interrupt Generator Logic

Example

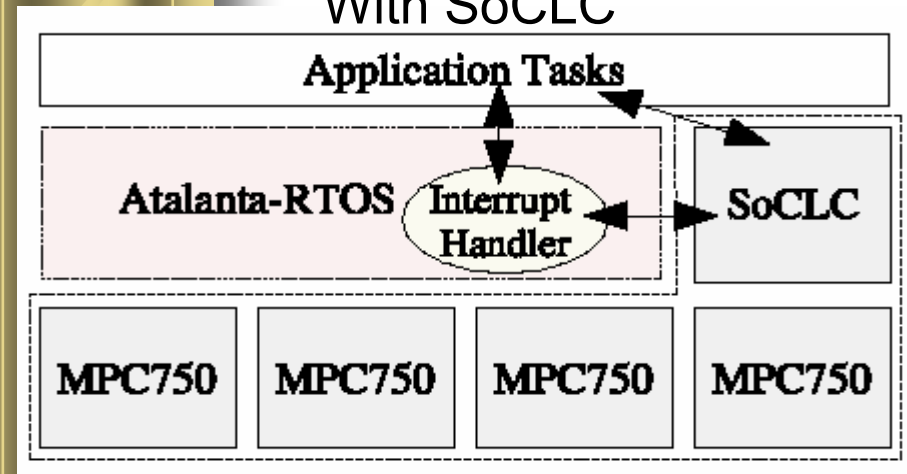

task1
task2
task3
E
CS1
CS2
E
E
CS2
E
E
CS1
Lock variables
States of blocked tasks
Priority of lock owner
Ceiling
Dynamic priority of tasks
1
2
0
1
0 0 0 … 0
0 0 0 … 0
0
2
1
1
1
1
2
3
1
1
3
Priority Encoder
register
Highest priority task
PE tasks
priority
Interrupt Generator Logic

Experimental HW/SW Architecture (1)

Without SoCLC



With SoCLC

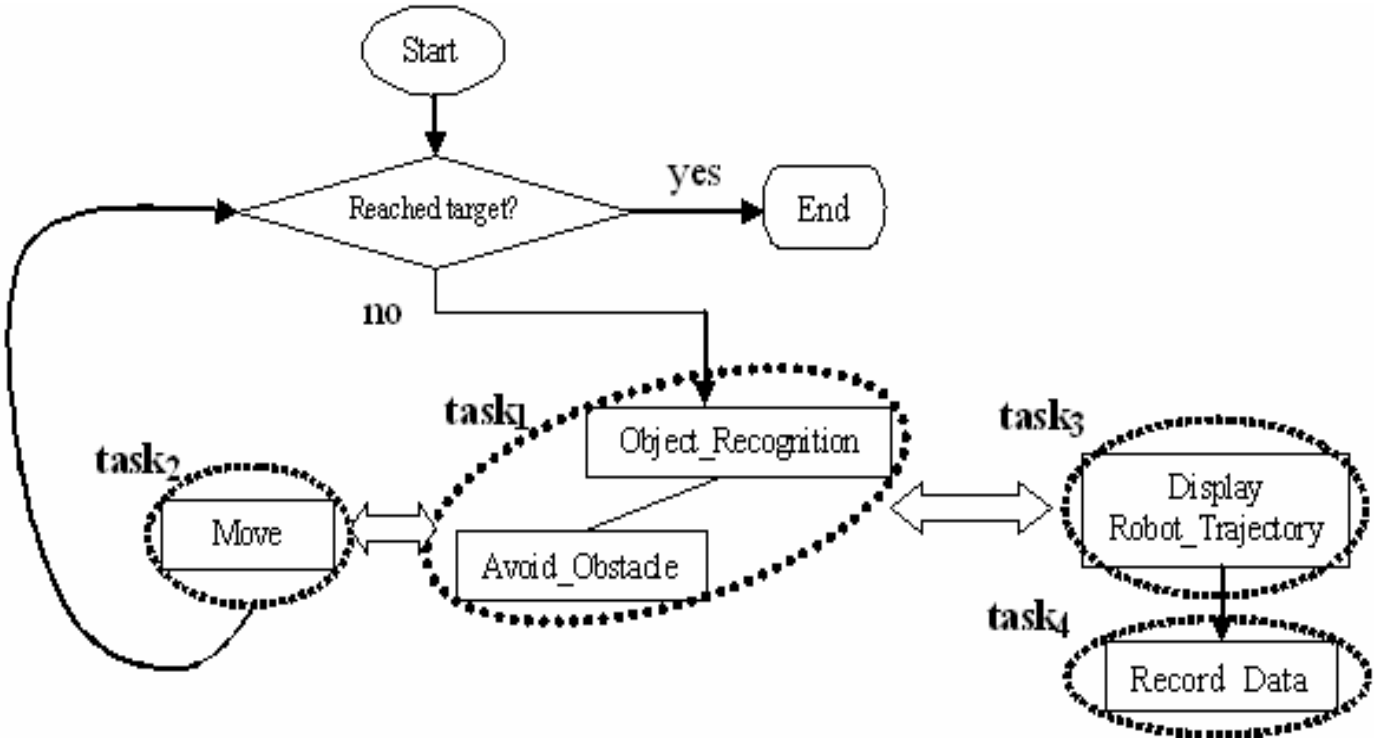


- Multiple application tasks and Atalanta-RTOS
- Multiprocessor setup with MPC750s on Seamless CVE (from Mentor Graphics)
- Atalanta-RTOS
- SoCLC provides lock synchronization among processing elements

Experimental HW/SW Architecture (2)

- MPC750 processors
  - 32 kB data cache
  - 32 kB instruction cache
  - 300 MHz internal clock speed
  - 100 MHz global bus system clock speed
- Shared memory size: 16 MB

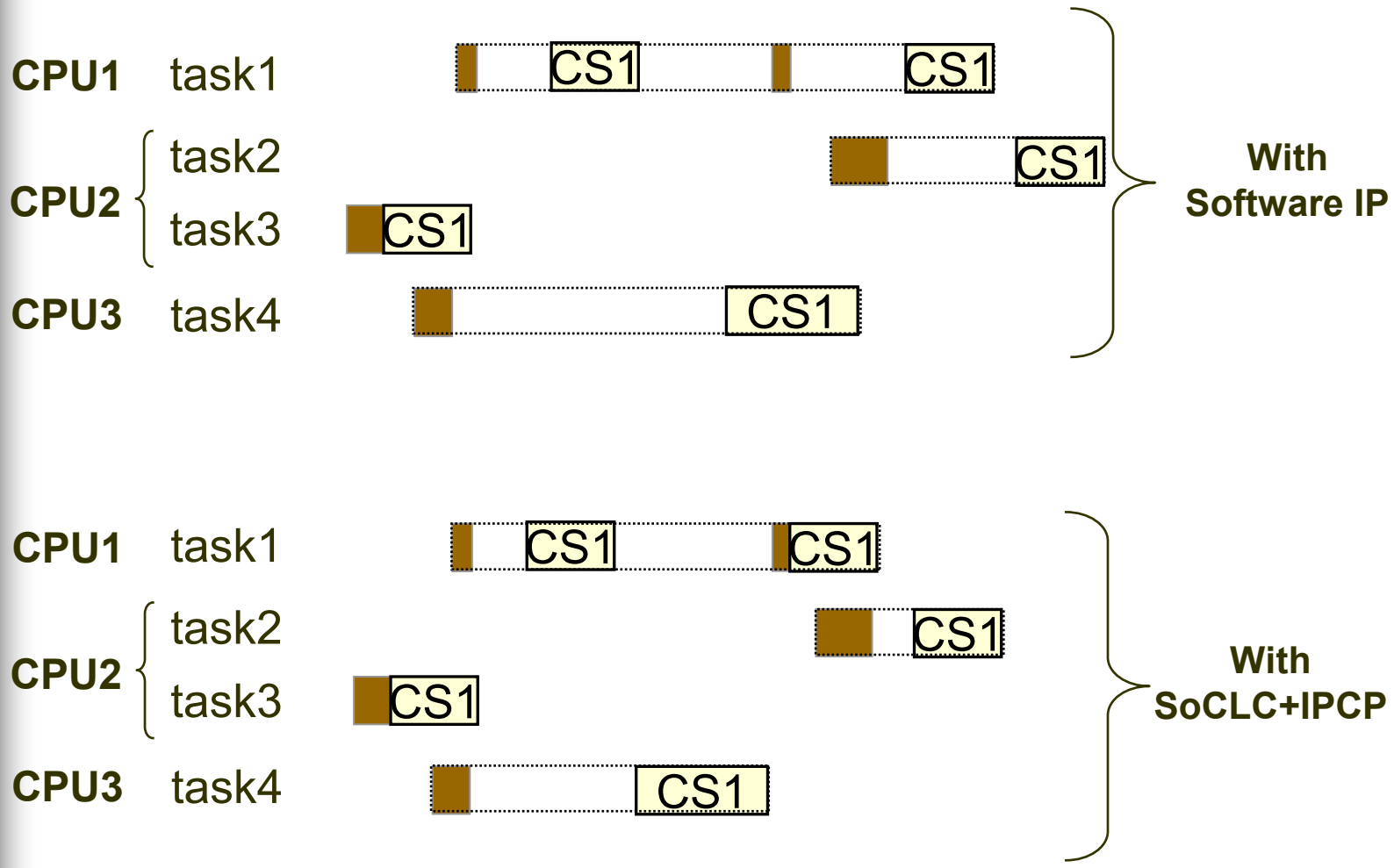
Simulation Scenario:
a robot application



Task Priorities

- Task1 → highest priority task with critical hard real-time requirement (response time: 250 us)
- Task2 → second highest priority task (response time: 300 us)
- Task3 → third highest priority task (response time: 300 us)
- Task4 → lowest priority task (response time: 600 us)

Execution Trace



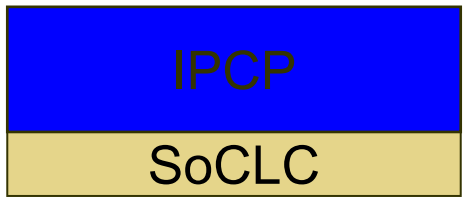
Experimental Results

	Task 1	Task 2	Task 3	Task 4
WCRT	250 us	300 us	300 us	600 us
Completion time for Software PI Protocol	283 us	556 us	80 us	517 us
Completion time for LCPI	93 us	247 us	77 us	337 us

	Without SoCLC	With SoCLC	Speedup
Lock Latency (clk cycles)	570	318	1.79 X
Lock Delay (clk cycles)	6701	3834	1.75 X
Overall Execution (clk cycles)	112170	78226	1.43 X

Synthesis Results

Number of processors	short CS locks	long CS locks	total number of locks	SoCLC with IPCP logic area	SoCLC without IPCP logic area	IPCP hardware logic area
4	16	16	32	5578	1694	3884
4	16	32	48	8690	2071	6619
4	32	32	64	8957	2329	6628
4	32	64	96	15263	3323	11940
4	64	64	128	15785	3740	12045



12045 NAND gate equiv.

3740 NAND gate equiv.

- Area in NAND gate equivalents in .25TSMC
- Can easily fit into on-chip eFPGA

Conclusion

- SoCLC: Custom hardware logic that improves lock-based synchronization performance in a multiprocessor SoC
- Priority inheritance support with SoCLC hardware
- Improves real-time predictability of the system and helps the application deadlines to be met